UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

HARVARD
MEDICAL SCHOOL

Master Thesis

Quantification of longitudinal tumor changes using PET imaging in 3D Slicer

Paul Mercea

Submitted to the Department of Medical Informatics in partial fulfillment of the requirements for the degree of
Master of Science (M.Sc.) Medical Informatics
at the
Ruperto Carola University of Heidelberg, Germany

Submitted:	July 2013
Presented by:	Paul Mercea born: 10.05.1985
Referees:	Prof. Dr.-Ing. Hartmut Dickhaus University of Heidelberg
	Prof. Ron Kikinis, MD Harvard Medical School

Surgical Planning Laboratory

BRIGHAM AND
WOMEN'S HOSPITAL
A Teaching Affiliate of Harvard Medical School

Declaration

I hereby certify that I have written this thesis independently, solely based on the literature and other sources listed in the bibliography and properly cited in the text.

Matriculation no. University of Heidelberg: 2664607
Matriculation no. Heilbronn University: 176850

Heidelberg, 19.07.2013 ….……...……...

Paul Mercea

Abstract

Quantitative assessment of Positron Emission Tomography (PET) imaging can be used for diagnosis and staging of tumors and monitoring of response in cancer treatment. In clinical practice, PET analysis is based on normalized indices such as those based on the Standardized Uptake Value (SUV). Although largely evaluated, these indices are considered quite unstable mainly because of the simplicity of their experimental protocol. Development and validation of more sophisticated methods for the purposes of clinical research require a common open platform that can be used both for prototyping and sharing of the analysis methods, and for their evaluation by clinical users. This work was motivated by the lack of such platform for longitudinal quantitative PET analysis.

By following a prototype driven software development approach, an open source tool for quantitative analysis of tumor changes based on multi-study PET image data has been implemented. As a platform for this work, 3D Slicer 4, a free open source software application for medical image computing has been chosen. For the analysis and quantification of PET data, the implemented software tool guides the user through a series of workflow steps.

In addition to the implementation of a guided workflow, the software was made extensible by integration of interfaces for the enhancement of segmentation and PET quantification algorithms. By offering extensibility, the PET analysis software tool was transformed into a platform suitable for prototyping and development of PET-specific segmentation and quantification methods.

The accuracy, efficiency and usability of the platform were evaluated in reproducibility and usability studies. The results achieved in these studies demonstrate that the implemented longitudinal PET analysis software tool fulfills all requirements for the basic quantification of tumors in PET imaging and at the same time provides an efficient and easy to use workflow. Furthermore, it can function as a platform for prototyping of PET-specific segmentation and quantification methods, which in the future can be incorporated in the workflow.

Preface

This work is the documentation of my master's thesis project at the Surgical Planning Laboratory, a teaching affiliate of Harvard Medical School, located at the Brigham and Women's Hospital in Boston, Massachusetts, USA. It was written for the Institute of Medical Biometry and Informatics at the University of Heidelberg Germany.

Acknowledgements

First of all, I would like to thank Prof. Dr. Ron Kikinis for inviting me to conduct my masters research at the Surgical Planning Laboratory in Boston and Prof. Dr.-Ing. Hartmut Dickhaus for offering me the possibility to take this opportunity.

Thanks to Steve Pieper for his refreshing ideas and his helpful advises in the software implementation part of this work. Thanks to Tina Kapur for her commitment to enforce the team spirit by organizing social events.

Special thanks to Dr. Andriy Fedorov for being a great supervisor during my visit in Boston and for supporting me with plenty of suggestions, great ideas and explanations and for teaching me valuable lessons about the work as a researcher.

I am incredibly grateful for the persistent support and everlasting motivation by my parents, Dorina and Peter Mercea, and my sister Petra Mercea, who were always there for me. Very special thanks to my grandmother, Judith Miess, for her moral and financial support throughout my entire studies. I wish to express my deepest gratitude to Inga Hahn for her unconditional support during my time abroad and for always believing in me.

Finally, I would like to acknowledge the University of Applied Sciences Heilbronn and especially the Baden-Württemberg Stiftung for making the research presented in this work possible by supporting my visit overseas with a Baden-Württemberg Stipendium scholarship.

Contents

List of Figures

List of Tables

Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
CLI	Command Line Interface (module type in Slicer)
CT	X-Ray Computed Tomography
CTK	The Common Toolkit
DICOM	Digital Imaging and Communications in Medicine
GUI	Graphical User Interface
ITK	The Insight Segmentation and Registration Toolkit
MRI	Magnetic Resonance Imaging
MRML	Medical Reality Markup Language
MVC	Model-View-Controller
NA-MIC	National Alliance for Medical Image Computing
PACS	Picture Archiving and Communication System
PET	Positron Emission Tomography
ROI	Region of Interest
SUV	Standardized Uptake Value
UML	Unified Modeling Language
VOI	Volume of Interest
VTK	The Visualization Toolkit
XML	Extensible Markup Language

1 Introduction

Positron emission tomography (PET) has become an essential tool for the routine diagnosis, staging and monitoring of tumors in oncology. Although its underlying concept is relatively simple, more than 30 years of development and specialization were required until it finally found it's way out of the research laboratories into clinical practice. Ever since, its importance within the field of clinical oncology has significantly increased, not least because of its possibility to indicate malicious cell growth, where other imaging modalities reach their limits. PET is a functional imaging technique with the focus on detection of physiological activities. However, the drawback of this technique is a modest anatomical imaging resolution. On the other hand, anatomical and morphological imaging techniques like X-ray Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), which are widely used in oncology, offer high anatomical resolution but are not capable of imaging metabolic activity. As a consequence, efforts were made to combine PET and CT scanners in order to benefit from the advantages of both techniques and recently also the combination of PET and MRI has been brought to market maturity.

In fact, PET is increasingly being used in cancer management but its potential regarding quantitative assessment remains widely underutilized in clinical practice, where diagnosis and staging of tumors are done either by visual inspection only or with additional help of radiology workstation software tools. Quantification of PET image data with these tools is almost exclusively done by application of the standardized uptake value (SUV) indices [1]. The prevalence of SUV is based on its simple experimental protocol, which allows quantification in a timely manner. The stability and applicability of the SUV method in all its implementation forms has been discussed extensively in literature (e.g. [2], [3], [4], [5], [6], [7]). Despite this, to date there still does not exist a standardized protocol for the application of SUV. Nonetheless, the benefits of using the SUV for quantification in clinical routine are currently bigger than its pitfalls [8]. More sophisticated methods for PET analysis based on dynamic imaging have also been developed in recent years [9], [10], [11]. However, these methods require prolonged times of successive scanning with the consequence of decrease in volume possible to scan. Considering these restrictions, said methods are not yet suitable for application in a clinical environment.

With efficient but relatively unstable indices on the one hand and superior but too sophisticated methods on the other, research in PET quantification has to aim for the improvement of existing tools and also for the development of new ones. The aim is to develop more reliable and at the same time effective techniques, and always leverage new possibilities gained through technical progress. Thus, the development of new

quantification algorithms and platforms for prototyping of those is required. These platforms must function as a basis, offering PET imaging and analysis functionality, and provide the possibility to be extended with new algorithms. Regarding these requirements, commercial PET imaging systems are usually not appropriate for the purpose of research and development since they often are neither extendable nor do they support prototyping of new functionality. In contrast to commercial solutions, open-source biomedical imaging software is especially developed for use in the academic environment. Mostly publicly funded and implemented by research groups from the field of biological or medical imaging, some of the freely available, open-source, non-restrictive software frameworks have the capability to provide state-of-the-art medical image visualization. Such frameworks ideally come with a decent selection of basic image analysis tools and represent the fertile soil for the development and prototyping of new algorithms for biomedical applications. From the various free and open source software tools, available for medical image visualization and processing, we have chosen 3D Slicer[1] as the software platform for our project. Besides its impressive visualization techniques of two, three and even four-dimensional (with time as the forth dimension) data, versatile functionalities are provided by over 100 different modules targeting almost every existing area in the field of medical image computing. With support of the Digital Imaging and Communications in Medicine (DICOM) format, advanced registration algorithms, easy to use segmentation tools and the possibility of chart plotting, 3D Slicer offers all the tools required for a promising implementation of our work.

The objective of this project is the development and evaluation of an open source software tool for quantitative analysis of tumor changes from PET image data with special focus on an elaborated workflow, meeting usability expectations of expert but also non-expert users in the field of PET image analysis. An additional goal of this project is to realize the implementation in a way that promotes extendibility of the developed tool. This makes it suitable for other algorithm developers who are or will be involved in the development and prototyping of PET-specific segmentation or quantification methods. The following steps describe the activities that have been executed during the development of this project.

First, a prototype driven requirements analysis was conducted in order to gather as many requirements as possible from potential users, developers but also clinical researchers. Second, the software architecture was designed with due regard to the optimal arrangement of the data structure in order to avoid complications in the subsequent implementation process and to streamline the same. In this project, particular relevance was paid to the usability of the resulting tool and a well-engineered

[1] 3D Slicer: http://slicer.org/

workflow. To achieve this goal, feedback from clinical and software engineering experts was collected throughout the whole development process in order to ensure that the orientation of the implementation is based on the requirements of the target user group.

The presentation of the work done and results obtained in this project is given as follows. In the next chapter, a brief overview of the underlying physical principles and image acquisition techniques of PET is given. Then, fundamental image analysis methods and the software frameworks and libraries involved in this work are shortly described together with the concept of usability engineering and its evaluation methods. In the following chapter, the methods that were used to implement and validate the software tool are outlined. Interim results in form of software prototypes and final results from the validation studies are presented in the penultimate chapter. Finally, the last chapter discusses the result achieved in this project and provides an outlook on the potential of further development of this work.

2 Fundamentals

Ever since its early days, the significance of medical imaging in diagnosis and therapy has rapidly increased. Above all, this is due to the explosive development in quantity, quality and diversity of medical imaging data and the benefits coming from the continuous evolution of software and hardware used for analysis.

In this chapter, the fundamental principles of PET imaging as well as the standardized protocol for medical imaging, fundamental image analysis methods, software toolkits and implementation guidelines used for implementation of the software in this project are presented.

2.1 Positron Emission Tomography

PET in clinical routine is a sophisticated procedure, which is performed by trained personnel following a precise schedule. In the following, an overview of the basic principles of radioactive tracing and positron emission and detection is presented. This overview is mainly based on the works of Jadvar [12], Lynch [13] and Martinez et al. [14]. Furthermore, current PET scanner configurations and methods for quantification of PET data are outlined.

2.1.1 Radioactive Tracing

In preparation for a PET scan, a radioactive labeled chemical compound (radioactive tracer) has to be produced in a cyclotron. Due to the relatively short radioactive half-life of radioisotopes suitable for PET imaging, the cyclotron is required to be located on site. The widest used radiotracer in current clinical routine is a fluorine-18 (^{18}F) isotope labeled glucose (^{18}F-FDG – Fluorodeoxyglucose). Subsequently to its production, this radioactive tracer is applied to the patient by intravenous injection. To ensure a sufficient distribution of the tracer inside the body before the image acquisition, an appropriate resting time is required but cannot take too long in regard to the radioactive decay of the radioisotope. The radioactive tracer ^{18}F-FDG is taken up and phosphorylated by the human body in the same way as regular non-radioactive glucose. However, instead of being converted into energy or stored as glycogen, ^{18}F-FDG is not undergoing any further reaction and remains trapped inside the cells where it gets accumulated. Given the fact that malignant tumor cells have a higher growth rate than benign ones, it is apparent that higher ^{18}F-FDG uptake will take place in tumors. By using a PET scanner, radioactive decay of the radioisotopes trapped inside the human body can be registered. Thus, the location of tumors in PET images can be estimated

due to the fact that tumor cells generate a more intense signal when compared to healthy ones [13].

2.1.2 Positron Emission

Positrons are basically positively charged electrons (antielectrons), produced by unstable radioisotopes with an excess of protons. Based on their positron excess, positron-emitting radioisotopes are neutron-deficient and gain stability through the transmutation of a proton (P^+) into a neutron (N). As a part of this process, a positron (e^+) and an electron neutrino (v) are emitted while the excess energy (E) of the decay (kinetic energy) is shared between the positron and the neutrino in different amounts [14]:

$$P^+ \rightarrow N + e^+ + v + E \tag{2.1}$$

Outside the nucleus, the positron is highly unstable and follows a jagged path while it is losing its kinetic energy by ionizing surrounding molecules when seeking to combine with an electron (e^-). This combination results in a positronium ion which is very short lived because the antimatter positron and the electron annihilate each other (within 2 nanoseconds) [13]. As a product of the annihilation process, two gamma rays (γ) are emitted in nearly opposite directions. Each one of these rays has an energy of 511 keV, resulting from the annihilation of the resting masses of the positron and the electron ($E = mc^2$):

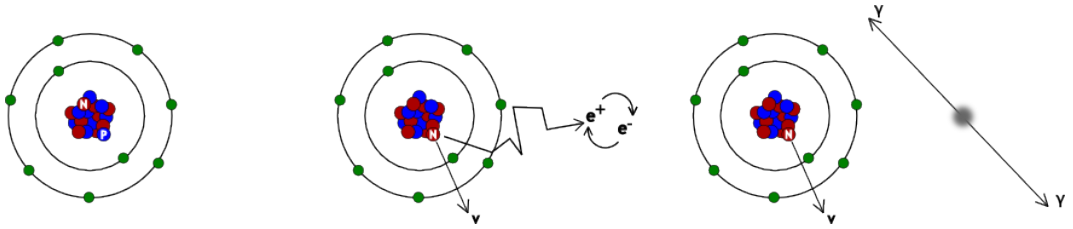


Fig. 2.1: Illustration of a radioisotope with positron excess emitting a neutrino (v) and a positron (e+), which after a short distance travelled, interacts with an electron (e-) resulting in an annihilation with emission of two gamma rays (γ).

2.1.3 Coincidence Detection and Imaging

The almost back-to-back emitted gamma rays (photons) from the positronium annihilation play the key role in PET. When those rays are detected at approximately the same time by two different detectors on opposite sides of a detector ring, the detectors are „in coincidence“ which means that the annihilation of the positronium must have taken place somewhere in between them on a so called line-of-response (LOR) [12]. Provided that the energy of the detected photons is above a certain lower

level of 350-400keV, they are analyzed by a coincidence-processing unit. Signals from opposite detectors that have been detected within a timespan of 4 – 12ns (coincidence window) are registered as a true coincidence event [14].

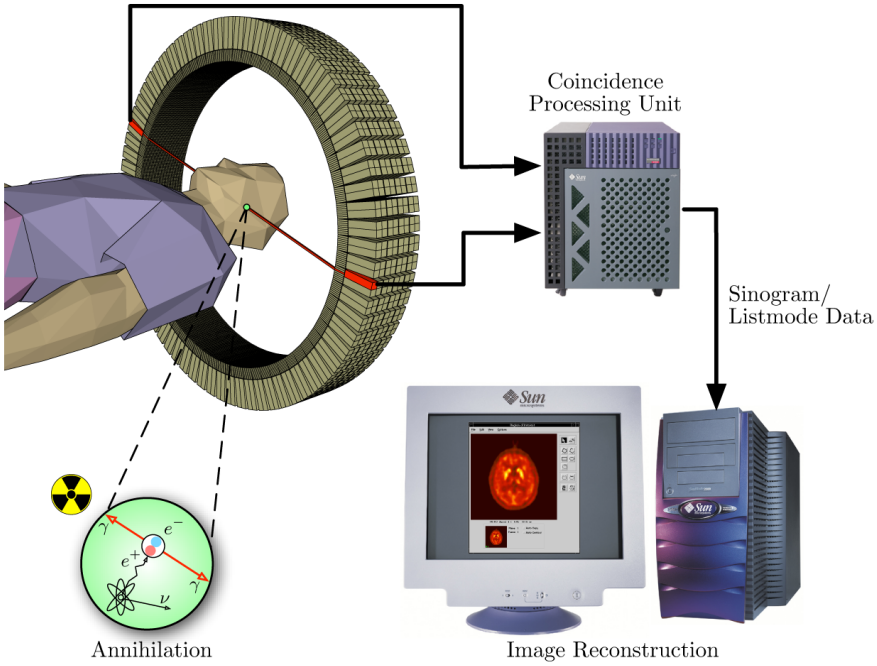


Fig. 2.2: PET scanner coincidence detection. Photons, emitted back-to-back during annihilation, are detected on opposite sides of the detector ring and classified as true coincidence event by the coincidence-processing unit. With the sum of all recorded coincidence events, a sinogram can be generated. From this sinogram, a volumetric image is then reconstructed in the end [15].

However, besides true coincidence events, scattered and random coincidence events also exist and are registered too [12]. A scattered coincidence occurs if one or both of the photons undergo scattering in the patient's body. In contrast to a non-scattered one, a scattered photon has less energy and changes the direction of its initial path. Assuming that such photons still have enough energy to be detected and these detections happen to be within the coincidence window, a coincidence event will be recorded although the line-of-response will be displaced. Random coincidence events occur when only a single annihilation photon reaches a detector while the other loses too much energy to be detected. Said case may occur due to attenuation and other scanner-related limitations. Assuming that two single photon detections occur within the coincidence window, these detections will be recorded as a true coincidence although they occurred independently. Recording of both scatter and random coincidence events results in loss of anatomical resolution in the resulting PET image.

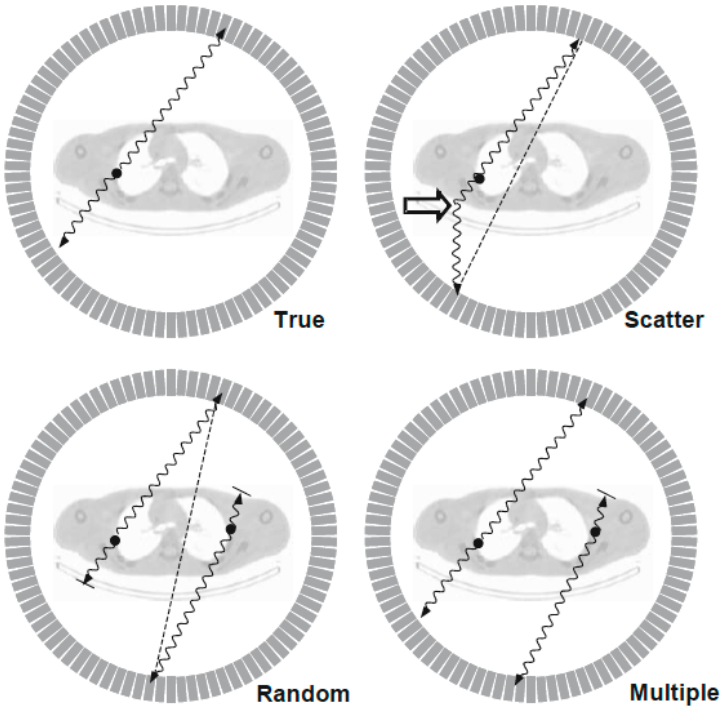


Fig. 2.3: Different types of coincidence detections in a ring scanner. The black dot indicates the location of positron annihilation. From top left clockwise: a true coincidence, a scattered coincidence with one photon leaving its path, multiple coincidence with two positron annihilations and three counted events and a random coincidence with two annihilations and one counted event for each [16].

Based on the count rate of registered coincidences for each detector pair, line integrals can be modeled forming a sinogram (analogous to CT imaging) which then can be transformed into three dimensional (3D) image data using specialized image reconstruction algorithms [17]. Although image reconstruction is a crucial step in the generation of PET image data, it is beyond the scope of this work to provide further details in this respect. Eventually, the recorded radioactivity (Bq/ml) is stored as scalar values representing the different grayscale intensities of the reconstructed 3D image.

2.1.4 Combined PET and CT

Compared to CT images, PET images show a lower anatomical resolution, which can affect the correct localization of lesions and demarcation of their borders. Moreover, the contrast in CT imaging is based on the density of the tissue, while contrast in PET imaging results from metabolical activity in the tissue. As a consequence, combination of PET and CT imaging allows leverage of both imaging modalities' advantages and can provide a tool for accurate localization of functional abnormalities. However, such combination requires a decent alignment of the PET and CT image data. In image processing, such alignment is called registration.

Image data is organized in a discrete coordinate system containing numerous elements. In two-dimensional (2D) images these elements are called pixels and in 3D image volumes they are called voxels. When acquired with standalone scanners, the coordinate systems of PET and CT image data will most certainly differ from each other. The reason for the difference can be manifold but the highest influence derives from change of patient position between the scans. It is practically almost impossible for a patient to maintain the exact same position when transferred from one scanner to the other.

The goal of the registration process is to compensate these circumstances by using software algorithms to transform one image data set into the coordinate system of the other. Nevertheless, minimization of patient motion between scans is a key factor for a successful registration. To cope with this requirement, the combination of PET and CT scanners into a single device has been realized in the early 2000s [18]. In contrast to standalone PET and CT scanners, such combined scanners represent an effective approach for the acquisition of accurately registered images and are able to contribute to the reduction of overall scan time by up to 40% [14]. The reason for this significant improvement is based on the fact that the scans are acquired subsequently thus minimizing the possibility of movement. Moreover, the patient is moved on an automatic bed from one scanner to the other describing a linear, one-dimensional translation that can be registered very fast and accurate by software algorithms.

2.1.5 Quantification of PET Data

Diagnosis and staging of tumors in PET imaging can be achieved through visual inspection by an evaluator with expert knowledge in oncology and anatomy. This approach is known as qualitative analysis. While it is a first-choice method for identification of tumors, it is only suited to a limited extent when it comes to prediction of treatment efficacy and assessment of treatment [8]. For such applications, quantitative analysis is more appropriate since it provides the required objectivity. In the context of this work, quantification describes the expression of imaged radioactive tracer uptake in a measureable entity.

Several sophisticated quantitative methods for PET quantification [9], [10], [11], [19], [20] have been presented over the past two decades. However, such methods have a complex protocol and require a prolonged data acquisition procedure. Because of that, these methods have not found their way into clinical routine yet. On the other hand, simpler and less exact quantification methods, so-called semi quantitative indices, for PET imagings are widely used in clinical practice today. The most accepted of those indices is the Standardized Uptake Value (SUV) [1] which is easily applicable and describes the ratio between the concentration of the injected radiotracer in a volume of interest (VOI) and the injected activity (corrected for physical decay) normalized by a normalization factor:

$$SUV = \frac{radiotracer\ concentration\ (\frac{kBq}{ml})}{\frac{injected\ activity\ (MBq)}{normalization\ factor}} \tag{2.2}$$

By using a normalization factor it is taken into account that the distribution of radioactive tracer in the human body is strongly influenced by physique. Commonly used normalization factors are body weight (bw), body surface area (bsm) and lean body mass (lbm). While for the computation of body weight normalized SUV only the patient's body weight at the time of image acquisition is required (besides concentration and injected activity), the body surface area has to be determined based on weight and height of the patient by using approximation equations [6]. The lean body mass method requires estimation of the wrist, waist and hip circumferences in addition to the patient's body weight [3].

On the one hand, the simplicity of the SUV indices makes them suitable for use in clinical routine, while on the other hand this simplicity is causing instability of the results. A strong positive correlation between body weight normalized SUV and the body weight has been found [4], leading to overestimation of uptake in obese patients. Additionally, both the weight-independent body surface area normalized SUV and the lean body mass normalized SUV computation methods are also not specific enough to allow comparability in multi center studies. This is because neither the time interval between radiotracer injection and image acquisition nor scanner specific properties are taken into account. Moreover, the SUV indices are less suitable for quantification of PET data that was acquired using radioactive tracers with a distribution in the human body differing from that of ^{18}F-FDG. Thus, the conclusion is that, for the quantification of PET imaging acquired with novel radioactive tracers, quantification methods with different kinetic properties will be required in the near future [8].

2.2 The DICOM Standard

In 1983, the American College of Radiology (ACR) and the National Electric Manufacturers Association (NEMA) formed a joint committee to develop a standard that would make communication in digital medical imaging independent of device manufacturer in order to facilitate the development and expansion of picture archiving and communication systems (PACS) [21]. Currently in its third version cycle, the standard for Digital Imaging and Communications in Medicine (DICOM) consists of 18 parts (from 1-20, part 9 and 13 being retired) and had its last publicly available revision in 2011. It is free and can be obtained from the official DICOM home page[2], which is maintained by the NEMA.

In digital medical imaging, DICOM provides all the required tools for the diagnostically accurate representation and processing of collected data. By being an all-encompassing data transfer, storage and display protocol, DICOM is well-suited to cover all functional aspects of digital medical imaging [22].

The following two paragraphs give a brief overview of the structure of DICOM data and the hierarchy that is used to organize it.

2.2.1 DICOM Information Model

The main challenge for the DICOM standard is to provide an infrastructure capable of organizing the complex medical environment in a precise and device-independent way. By using its own lingo, the DICOM standard specifies a set of Information Object Definitions (IODs) which provide an abstract definition of real-world objects applicable to communication of digital medical information [23, p. 46]. In DICOM, all real-world data (patients, medical devices, images, etc.) are viewed as objects with respective attributes. For example, a patient IOD could consist of attributes like name, ID number, sex, age, size, weight, and so on. The transition from a real world patient to a patient IOD is illustrated in Fig. 2.4. IODs represent collections of attributes and in general contain as many of them as required to describe all clinically relevant information of a real world object [22]. The totality of these attributes form the DICOM Data Dictionary, which is listed in the sixth part of the standard [24]. The dictionary consists of over 2000 data attributes and ensures the consistency in naming and processing of those. Once medical data is stored as DICOM Data Attribute it can be transmitted and processed by various DICOM conform devices.

[2] DICOM Standard: http://medical.nema.org

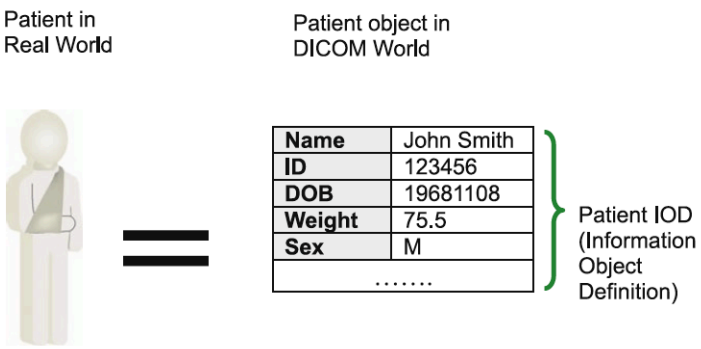


Fig. 2.4: Representation of real world data as DICOM Information Object Definitions (IODs) [22].

Since clinical data comes in a wide variety of formats, the standard defines 27 basic data types called value representations (VRs) [25]. Each value representation is characterized by its two-character name, repertoire of allowed characters (for alphanumerical values) and the maximum or fixed length of its actual value in bytes. Depending on the type, data can be stored either as text or as binary representation. While the text format is well suited for names, dates and other character strings, the binary format is the first choice when it comes to encoding of single numerical values or sequences, as is the case for image pixels.

Although this paragraph only provides a very brief overview of the DICOM information model, one can easily understand the enormous potential it bears for precise modeling of real world medical objects and the standardized exchange of captured data.

2.2.2 DICOM Information Hierarchy

In the DICOM standard, organization of the numerous attributes from the data dictionary is realized with a four level DICOM information hierarchy. From top to bottom these levels are *Patient*, *Study*, *Series* and *Image* (see Fig. 2.5). Each parent node in the hierarchy can have one or multiple child nodes. For example, a patient may have multiple studies when he has to undergo follow-up imaging for assessment of treatment response. A study in turn may consist of one or more image series. The latter is the case for multi-modality imaging techniques like PET/CT, in which image series from both modalities are captured during one study. Eventually, each image series may also contain more than one image.

In the implementation of the hierarchy, a key-level ID is assigned to each level. At the *Patient* level, the “Patient ID” is used. It usually has the objective to identify a patient uniquely in the system of an institution. On the level below, each study is assigned a unique identifier called (UID) the “Study Instance UID”. The same applies for the *Series* level where “Series Instance UIDs” are used to identify individual series.

At the bottom level, each image has its own “SOP Instance UID” [22]. All of the above mentioned unique identifiers are mandatory for any type of imaging with DICOM conform devices. When properly used, they can significantly improve data identification and facilitate hierarchical data searches, retrievals and transactions.

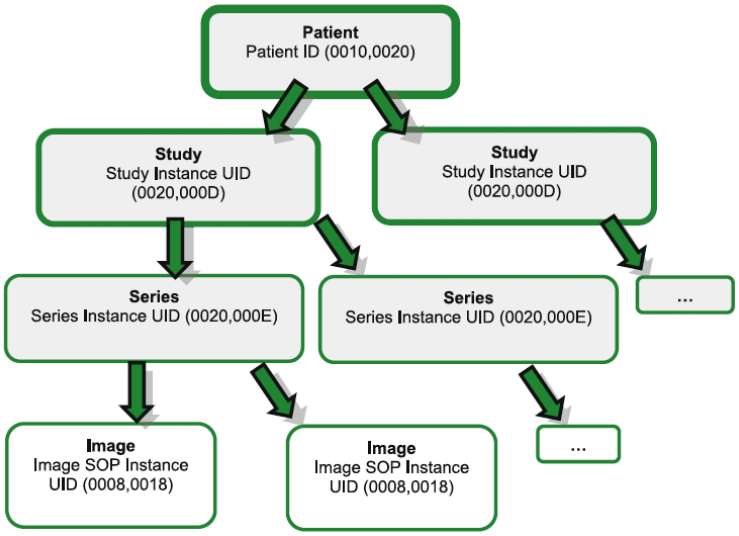


Fig. 2.5: Four-leveled DICOM Information Hierarchy with key elements (UIDs) for identification of each level [22].

2.3 Medical Image Analysis Techniques

In a general sense, the purpose of this work is to support the analysis of medical images. However, medical image analysis describes a broad field of numerous methods, reaching from rather universal ones, suitable for application in common use cases, to those designed for highly specific tasks. In view of this fact, the following sections provide a brief overview of some medical image analysis techniques used for the realization of this work.

2.3.1 Image Data Visualization

As a first step in medical image analysis, the binary stored image data has to be presented in a way that is comprehensible for the examiner. For 2D image data, medical imaging software follows the same principle as any software capable of displaying images. Thus, the intensity values of the image pixels, stored as binary values in the file, are displayed in a pixel grid on the monitor according to their order. Presentation possibilities for 3D image data (consisting of a series of 2D image slices) are twofold. One solution is to display its 2D images next to one another in a grid and in a consecutive order similar to printouts of 3D image data. The second solution is based on interaction of the examiner who has to navigate through the individual 2D image slices of the 3D image data and only the selected image is displayed at a time.

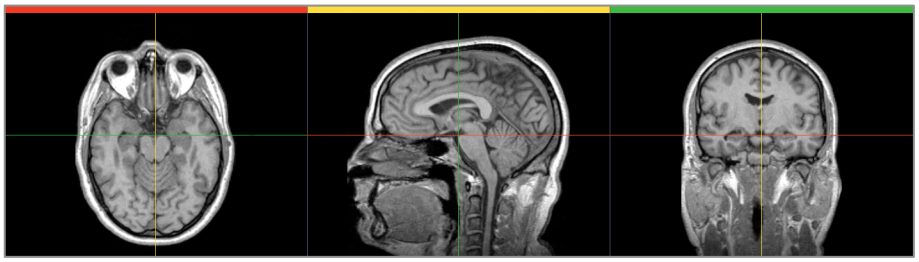


Fig. 2.6: Visualization of CT head data slices on axial (red), sagittal (yellow) and coronal (green) planes. The slice intersections are illustrated by the colored crosshairs.

A characteristic of 3D image data visualization is the possibility to display it on three different planes that are perpendicular to each other. In medical imaging applications these planes derive from the anatomical coordinate system. The *axial* plane separates the head from the feet while the *sagittal* plane separates left from right and the *coronal* plane separates front from back. Normally one of those planes corresponds to the acquisition plane (e.g. CT images are always acquired axially), so its visualized image data derives directly from the image file. However, the specific 2D image representation on the other two planes has to be calculated from the different 2D image slices before it

can be visualized. An example of how 3D image data is visualized on axial, sagittal and coronal planes is shown in Fig. 2.6.

2.3.2 Volume Rendering

In the recent years, volume rendering has become a very popular visualization technique for medical image data. On the one hand, this is due to the continuous improvement of computer hardware and especially graphics adapters. On the other hand, it is also its ability to display a discretely sampled 3D grayscale volume as a colorful, shiny and more anatomical object what makes volume rendering so valuable. Once started as an experimental technique in medical imaging, volume rendering has developed into a useful tool for clinical practice [26]. Its fields of applications are manifold as it can be used for noninvasive coronary imaging [27] as well as for surgery planning for living liver donation [28] and many more. A volume rendering projection from abdominal CT image data is shown in Fig. 2.7.

There exist several methods for obtaining a volume rendered 2D projection of a 3D data set. The most straightforward is called “ray casting”. This method is also used in this work for volume rendering of 3D PET image data. The basic principle behind ray casting is to determine the value of each pixel in the projection by sending a ray through the pixel into the scene and then evaluate the data encountered along the ray using a specific function (ray function) in order to compute the resulting pixel value [29]. For colored 3D volume rendering, an additional color transfer function is applied to estimate the color of the computed pixel values.

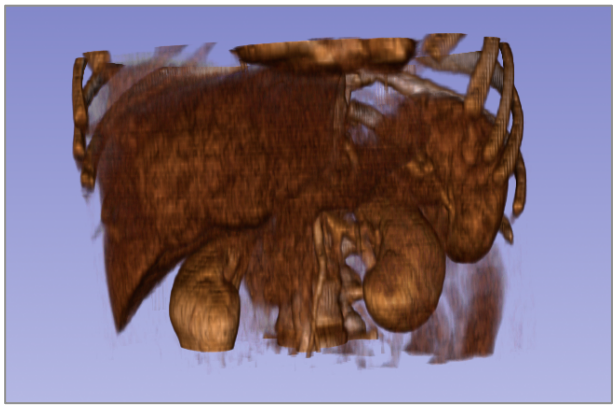


Fig. 2.7: Colored ray-cast volume rendering of the abdominal area computed from 3D CT image data.

In the case of volume rendering of 3D medical imaging, the scene consists of 2D image data slices representing a 3D image and the data encountered along the way are the scalar values of the voxels, which are used for computation in the ray function.

Volume rendering computation is a highly complex process and can be specialized by far more parameters than presented above. However, explanation of those is far beyond the scope of this basic introduction.

2.3.3 Registration

In medical imaging, registration describes the determination of a geometrical transformation which is able to align points from one image data set of an anatomical region of the body with corresponding points from another image data set of that same region. As already explained in section 2.1.4, these image data sets will most certainly differ from one another when not acquired subsequently with a single device. Differences can be due to variations in patient positioning, changes of patient anatomy (weight-loss, differently filled bladder, respiration, etc.) or varying acquisition settings of the scanning devices (e.g. CT is always acquired in the axial plane while the orientation of acquisition in MRI is arbitrary). However, for an accurate assessment of changes between different single-modality image data, or a precise overlay of images from different modalities, an appropriate alignment of those is worth striving for.

In general, image registration methods can be classified into three categories. Point-based methods work with a priori identification of corresponding point pairs in both images. Such points are called *fiducials* and are placed either at unique anatomical landmarks or are provided by artificial markers that have been implanted before acquisition of the images. When the required amount of fiducial pairs has been identified in both images (min. 3 pairs for 3D images), the registration method can estimate the geometric transformation that provides the best alignment. Surface-based methods use surface boundaries of 3D anatomical objects to determine corresponding surfaces in different images and calculate transforms for appropriate alignment. Finally, intensity-based methods estimate the highest similarity of two different images based on their voxel values alone. In the simplest form, an intensity-based registration method is iteratively optimizing a registration transform based on a similarity measure.

Registration transforms calculated by registration methods can be divided into two main categories. On the one hand, rigid transforms are geometrical transformations that preserve all distances, straightness of lines, planarity of surfaces and nonzero angles between straight lines (rigid transformations include translation, rotation and reflection and combination of those). On the other hand, non-rigid transforms include scaling, affine, projective, perspective and curved transformations [30].

2.3.4 Segmentation

Segmentation in medical imaging describes the process of partitioning an image into contentual coherent segments by classification of pixels/voxels according to specific criteria of homogeneity. In general, such segments correspond to different types of

tissues, organs or other task-related structures of interest. For classification, labels are assigned to each pixel/voxel of an image. Labels holding the same value classify a certain segment. For medical image analysis, segmentation is a useful technique since it allows the estimation of surface and volume of certain structures and can be used for generation of 3D surface models of the segmented objects, as shown in Fig. 2.8.

There are different approaches to segmentation, which in general can be characterized as manual, semi-automatic and automatic segmentation. An operator with expert knowledge in anatomy can perform manual segmentation by manually labeling the images. Hereto, manual segmentation tools, similar to the different shape and free hand painting tools in drawing software, can be used to apply the labels to the image data. In semi-automatic segmentation, software algorithms are used for the segmentation process but as the name of the method suggests, they require human interaction in order to perform correctly. In most cases, a starting point or region inside the object to be segmented has to be seeded or roughly outline so that the algorithm can iteratively refine the segmentation. Automatic segmentation methods are designed to perform without human interaction. They mostly rely on automatic contour and shape detection and use parameterized shape templates which are deformed to match the object to be segmented in the image [31], [32].

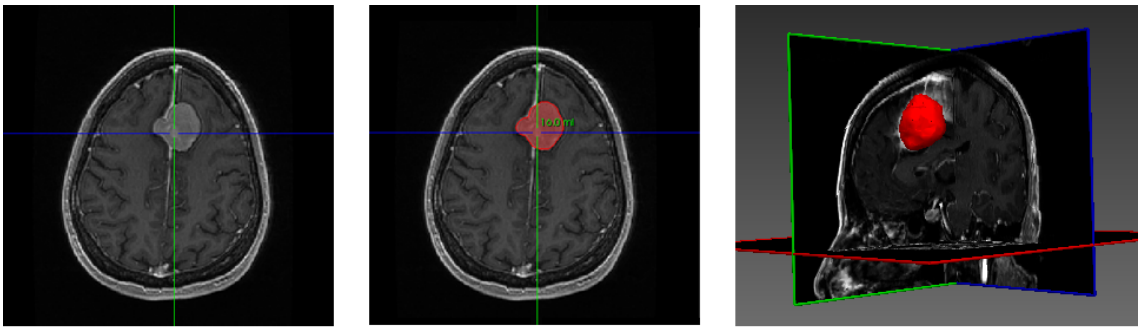


Fig. 2.8: Segmentation in *MITK 3M3*[3]: MRI head data with tumor (a); segmentation and volume assessment of tumor (b); 3D triangulated surface model of the segmented tumor (c).

Although a myriad of different segmentation methods and algorithms have been developed in the recent past, there is no single approach that can generally solve the problem of segmentation for the variety of existing imaging modalities. However, the most effective algorithms are obtained by customized combination of different components. By tuning the parameters of these components for the characteristics of the respective modality and features of the anatomical structure quite decent results can be obtained [33].

[3] MITK 3M3: http://mint-medical.de/productssolutions/mitk3m3/mitk3m3

2.4 The NA-MIC Kit

The National Alliance of Medical Image Computing (NA-MIC) is a multi institutional alliance in the United States, created with the objective for the implementation of an open-source infrastructure for the development and application of advanced imaging technologies for interdisciplinary research in the field of biomedical imaging [34]. The NA-MIC Kit[4] provides well-developed tools for algorithm developers, software engineers, and application domain scientists. These tools can be categorized into programming toolkits, end-user application software and system infrastructure [35] (see Fig. 2.9).

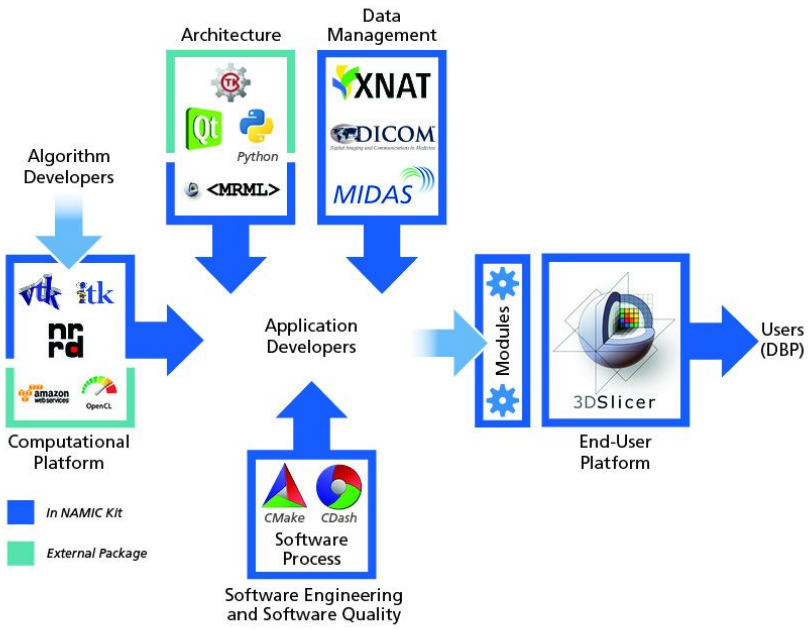


Fig. 2.9: NA-MIC-Kit software overview [36].

The majority of the toolkits used in this projects are elements of the NA-MIC Kit, while all others are external packages provided by the open-source community, meaning that the resulting software from this project will also be able to fulfill the open-source philosophy of the NA-MIC.

In the following, an introduction of the NA-MIC programming toolkits and the end user application software used in this project is presented.

[4] NA-MIC Kit: http://na-mic.org/Wiki/index.php/NA-MIC-Kit

2.4.1 Common Toolkit

The Common Toolkit (CTK)[5] is an open-source library for biomedical image computing, which in the words of its developers is: „developed with the objective to provide support code for medical image analysis, surgical navigation and other related projects“ [37]. Besides application-level DICOM support and specialized graphical user interface (GUI) widgets[6], CTK also offers a plugin framework and an automatic testing framework for the development of new software applications.

2.4.2 Visualization Toolkit

The Visualization Toolkit (VTK)[7] is an open-source, cross-platform development platform for 3D computer graphics, image processing and visualization [29]. Besides basic processing capabilities for image, surface and mesh-data, its main purpose is the visualization of 3D image data using basic and advanced visualization algorithms.

VTK's class library is implemented in C++ but also includes interfaces for automated wrapping techniques, thus it is also accessible from interpreted programming languages like Tcl, Pyhton and Java. The main characteristic of VTK is its event-driven architecture, in which the program flow is determined by the occurrence of events. Based on its sophisticated architecture, robust implementation and advanced image processing and visualization functionalities, VTK is being used in several bio-imaging software applications and is the basis of 3D Slicer, the software platform used in this project.

2.4.3 Insight Segmentation and Registration Toolkit

The Insight Segmentation and Registration Toolkit (ITK)[8] is an open-source, cross-platform development framework well-suited for medical imaging related tasks [33]. It provides algorithms especially for segmentation and registration of multidimensional image data and serves as a platform for the development of new algorithms in image analysis.

ITK is implemented in C++ with high emphasis on a generic programming style using C++ template code. Since the main advantage of generic code is that its functionality can be adapted to more than one data type, high modularity of the code and high reusability are ensured. Due to the high modularity, connections of several

---

[5] CTK: Common Toolkit, http://commontk.org

[6] Widget: Reusable element of a GUI

[7] VTK: Visualization Toolkit, http://www.vtk.org

[8] ITK: Insight Segmentation and Registration Toolkit, http://www.itk.org

ITK components can be composed to form sophisticated pipelines for the processing of complex image analysis algorithms.

2.4.4 Qt Libraries

The Qt libraries derive from an open-source, cross-platform application and user interface development framework. With more than 800 classes [38] written in C++, this framework provides an enormous breadth of functionalities suited for almost any conceivable task in software development. One of the most important characteristics of Qt is the fact that it makes use of a special code generator. It is included in the framework and called the *Meta Object Compiler* (*moc*) [39]. This generator is used for realization of Qt's *signal* and *slot* mechanism [40]. Hereby, a simple implementation of the *Observer* design pattern can be realized. Qt objects have the capability to emit specific Qt signals upon certain events (e.g. when a button is clicked). Then, other Qt objects with slot methods can receive these signals. Using the signal and slot construct helps to create a powerful, type-safe and flexible communication structure for the graphical user interface of an application.

2.4.5 3D Slicer

Currently in its forth version cycle, 3D Slicer (further referred to as Slicer) represents the end-user platform of the NA-MIC Kit. Like the other components, it is a free, open-source and cross-platform software package. Slicer's binary installer packages for Windows, Mac OSX and Linux platforms are available for download as well as its source-code, targeting developers and compilation on other systems like Oracle's Solaris.

DICOM standard [41] support, VTK based visualization possibilities for two, three and four dimensional image data, versatile processing and multimodal analysis functionalities and a streamlined graphical user interface (GUI) form Slicer into a powerful tool for medical image visualization and analysis while keeping the requirements for its operation so low that even a nontechnical user is able to easily operate the software.

The architectural concept of Slicer 4 describes a layer-based modular approach (see Fig. 2.10). The lowest layer contains platform specific hardware and OpenGL drivers for window management and rendering functionality provided by the host system. One level above, programming and scripting languages together with toolkit libraries provide higher-level functionality and abstraction for the upper level application and its extensions.

Regarding the application itself, Slicer's components can be categorized into the application core, Slicer modules and Slicer extensions. The core implements the user interface, data input and output functionalities and provides plug-in interfaces [42].

Modules represent the plug-ins for those interfaces. Being dependent on the application core, the modules' objective is enhancement of the core functionalities. In contrast to the modules, which are part of the application bundle, extensions are not part of the application's distribution. They function as external modules that can be integrated into the application „on-demand".

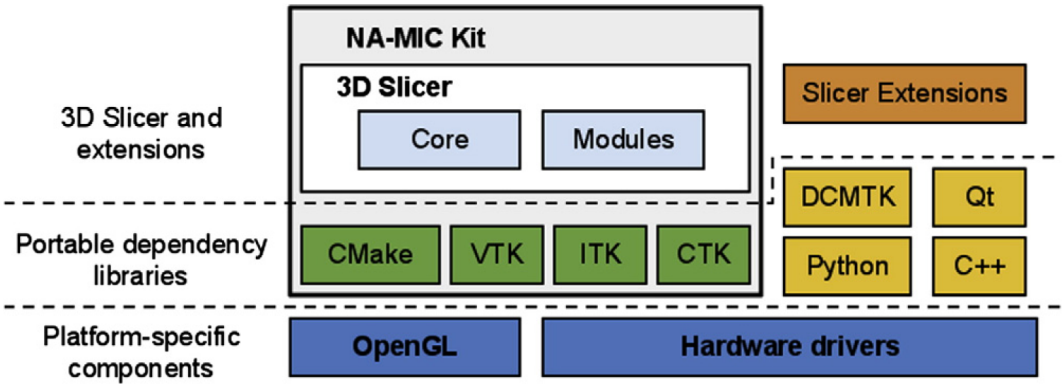


Fig. 2.10: System architecture and external library dependencies of Slicer 4 [42].

The core of Slicer 4 and the majority of the modules are implemented in C++. Due to support of the scripting language Python[9], so-called "scripted" modules have become a promising alternative to modules written in C++. Their implementation is in many cases more time-effective while drawbacks in computing performance are negligible. The classes implementing the application core follow the *Model View Controller* (MVC) design pattern, which requires a separation of the Model (data), View (GUI) and Controller (logic). Communication between those three components takes place either through direct access (Controller and View have access to Model) or through event handling mechanisms (Controller and View react to changes in Model), which are implemented following the *Observer* design pattern.

Data representation and organization in the Model is based on the Medical Reality Modeling Language (MRML) library which provides application programming interfaces (APIs) for the management of medical image data types, organization of those in a tree structure and serialization in an XML-based data format, which is suitable for exchange between different systems. Apart from the `MRMLScene`, which represents the root of the tree structure, objects from the MRML library are referred to as "nodes" since they all derive from the base class `MRMLNode`.

In Slicer 4, the GUI implementation is based on functionality provided by the Qt toolkit libraries[10] and on GUI widgets specialized for bio-medical imaging tasks, available from the CTK library. The processing functionality in 3D Slicer is

[9] Python Programming Language: http://python.org

[10] Qt: http://qt.digia.com

encapsulated in the logic, where computation with data from the `MRMLScene` is performed. Changes of the data structure in turn induce updates of the GUI elements presenting the data.

The ongoing maintenance and development of 3D Slicer is carried out by a community of more than 80 authorized developers worldwide [42] and supported by a user community that can report issues on open mailing lists or bug tracking systems. In addition to that, the stability of the software is tested on a daily basis for a variety of platforms. Results of these tests are summarized and reported online[11].

In the remainder of this topic an overview of the Slicer modules used for the implementation of this project is given.

2.4.5.1 DICOM Module

Introduced with Slicer 4, the DICOM module[12] represents the main user interface for import of DICOM data into the application. Imported DICOM data, either from disk or queried from a Picture Archiving and Communication System (PACS), is stored in an SQLite[13] based local database on the application's host platform. The easy to use DICOM browser GUI provides selection functionality and offers thumbnail previews of the available data sets. Moreover, the module is supporting extension through plug-ins which allows developers to implement new algorithms for customized DICOM data import (e.g. simultaneous loading of coherent PET and CT datasets).

2.4.5.2 PET SUV Image Maker Module

Belonging to the category of quantitative modules, the PET SUV Image Maker module is based on the Standard Uptake Value Computation module[14]. It provides functionality for the SUV based normalization of PET image volumes. Besides the captured radioactivity values stored in the pixel data of the image volume, additional information from the PET DICOM files is also required. When executed, the SUV_{bw} value for each voxel from the original image is computed and stored in the respective voxel of the output image volume.

---

[11] Slicer CDash project: http://slicer.cdash.org

[12] DICOM Module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/DICOM

[13] SQLite: http://sqlite.org/

[14] SUV Computation Module: http://slicer.org/slicerWiki/index.php/Documentation/4.0/Modules/PETStandardUptakeValueComputation

2.4.5.3 Volumes Module

Changing the appearance of volumetric image data in 3D Slicer requires functionalities from the Volumes module[15]. Besides the selection from a vast range of predefined lookup tables, modification of Window/Level settings as well as threshold functionality are available. Adjustment of image data plays an important role in PET/CT fusion imaging where application of a suitable colored lookup table to the PET image and task appropriate Window/Level settings can have a remarkable impact on the appearance of the fusion.

2.4.5.4 Volume Rendering Module

Fully interactive 3D visualization of volumetric image data can be initialized and adjusted using the Volume Rendering module[16]. As already mentioned in section 2.3.2, 3D volume rendering is a sophisticated technique. On the one hand it provides versatile visualization possibilities and on the one hand it requires high computational performance. For that reason, proper adjustment of the numerous parameters, involved in volume rendering computation, is very important in order to obtain adequate performance while maintaining reasonable hardware requirements. Considering these facts, the Volume Rendering module offers, amongst others, the selection of different CPU or GPU based volume-rendering techniques, interactive limitation of the rendered volume and various opacity and color mapping settings. Due to these possibilities, advanced volume rendering of complex 3D structures can be made available even on non high-end computation platforms.

2.4.5.5 Editor Module

A variety of tools for manual or semi-automatic segmentation can be found in Slicer's Editor module[17]. The selection ranges from simple voxel based painting tools, over more advanced dilation and erosion effects, right up to sophisticated segmentation algorithms like the GrowCut [43]. Segmentations generated with the Editor module are stored in so-called label map volumes, which then can be used for postprocessing like generation of 3D triangulated surface models or quantitative analysis. Similar to the DICOM module, the architecture of the Editor module also supports extension by plug-ins. Developers can implement their own segmentation algorithms and easily integrate them for validation.

---

[15] Volumes Module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/Volumes

[16] Volume Rendering Module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/VolumeRendering

[17] Editor Module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/Editor

2.4.5.6 Model Maker Module

As mentioned above, segmented image data can be used to generate 3D triangulated surface models, wherefore the functionality of the Model Maker module[18] is required. Models, created with the Model Maker, can be displayed simultaneously with 3D volume rendered image data. This functionality allows advanced visualization of segmented volumes in an anatomical context.

2.4.5.7 Label Statistics Module

Label map volumes storing segmentation results can be used not only for surface model generation, but also for calculation of statistics for the volumes classified by the labels. In Slicer, the Label Statistics module[19] provides such functionality. Currently supported statistics are, the total amount of voxels in a segmented volume, its volumetric extent in mm^3 or cubic centimeters and the minimum, maximum, mean and standard deviation of the segmented voxel gray scale intensities. Moreover, the possibility of plotting these statistics by using Slicer's charting functionality or exporting them in a text file is also provided.

2.5 Usability Engineering

The increasing importance of computers in everyday life came along with increasing requirement of human-computer interaction. Study of such interaction is a main discipline in the scientific field of usability engineering, which is concerned with the development of human-computer interfaces with high usability. In general, usability of a hardware or software system refers to an effective and efficient accomplishment of tasks, performed with human interaction. Nielsen [44], one of the major usability engineering researchers named five attributes which can be regarded as quality components of usability [45]:

1. Learnability: How easy it is for new users to accomplish basic tasks the first time they encounter a new system?

2. Efficiency: Once users have learned the system, how quickly can they perform tasks?

[18] ModelMaker Module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/ModelMaker

[19] Label Statistics module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/LabelStatistics

3. Memorability: When users return to the system after a period of not using it, how easily can they reestablish proficiency?

4. Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?

5. Satisfaction: How pleasant is it to use system?

2.5.1 Biomedical Software Usability

Medical imaging has become the backbone for diagnosis in many medical disciplines and modern medicine cannot exist without it. The development from simple x-ray scanners, producing single images per scan, to sophisticated imaging modalities like CT, MRI, PET and others, which usually generate large amounts of digital images, has brought along the necessity for systems to post process those images. There are two main user groups for medical image analysis systems. On the one hand, radiologists and specialized medical personnel use commercial analysis workstation as approved clinical devices for diagnosis and staging of diseases and assessment of treatment response. Researchers, however, have different objectives and require more flexible solutions than the commercial ones. For that reason, numerous algorithms and several software packages for the visualization and analysis of medical image data have been implemented by academic groups and in many cases made freely available for research, private and sometimes even commercial use. While users of commercial systems are trained for the correct usage of the installed software, private users and researchers not involved in the development of the software often have to learn the correct usage by themselves. Furthermore, software created by research groups is often used only for their own work, and often no efforts are undertaken to share their software with other research groups. This leads to redundant efforts when other groups intend to implement similar or identical algorithms. To avoid this redundancy and the waste of effort, academic software developers should always aim for the implementation of software in such a manner that it is also usable for people outside their research group. To achieve this goal, certain criteria have to be met. Carpenter et al. [46] have defined key criteria for development of bio-imaging software, which of course can also be applied to medical imaging software, not at least because these are also considered good practice in general software development. In the sense of the authors, bio-medical imaging software should be:

- User-friendly: Non-experts should be able to use the software, meaning that detailed help and documentation are a necessity as well as availability of open mailing lists, forums and installation routines for different computer platforms.

- Developer-friendly: Open-source licensing, well implemented, up to date available source-code and detailed documentation are key factors for academical software development.

- Interoperable: Dividable into several degrees, interoperability can be either implemented quite simple by providing the ability to read certain file formats across different systems or can get up to the seamless integration of one's software package functionality into another.

- Modular: Software modules, designed to solve a specific problem, can be combined with modules solving other problems by implementation of the same interfaces. Modularity is a very powerful tool for the development of software with a high level of customization possibilities, thus enabling the applicability in different domains.

- Validated: Extensive testing is the foundation for stable software and especially in domains like biomedical imaging, where results can be crucial for further decisions, software has not only to be tested but results have to be validated against reference results. Hence, the distribution of input data and its test results can help the user to validate whether the software is working correctly.

It is understandable that detailed implementation of the above listed criteria might not be possible for every software development project and research group. However, consideration of these criteria and their implementation to a certain degree can certainly improve the usability of the software for outside-users and increase the availability of the algorithms to external developers.

2.5.2 Usability Evaluation

Improvement of a system's usability can only be realized by the analysis of its current status of usability. Hereby, usability flaws can be detected and counter-measures can be taken to eliminate them. Empirical methods for the measurement of usability can be categorized as follows:

- Timing: Regarding the time users need to perform predefined tasks, conclusions can be drawn regarding efficiency, learnability and memorability of a system.

- Counting: While counting the required keystrokes for the performance of a certain task can help to measure its efficiency, counting of problem occurrence during a human-computer interaction can deliver useful information about the error-proneness of a system or difficulties regarding its learnability.

- Polling: Questionnaires addressing usability factors can help to gather users' opinions on the usability of a system. This is a useful method for getting feedback about the subjective satisfaction. However, there is only little correlation to the objective performance.

2.5.2.1 Heuristic Evaluation

In addition to the described usability measurement techniques, additional identification of usability issues can be achieved by the usage of a usability engineering technique called „heuristic evaluation". It is cost-effective, easy to learn, easy to use and can be conducted with only a small set of evaluators, since the proportion of found usability problems to number of evaluators stagnates relatively quickly (see Fig. 2.11). In order to perform a heuristic evaluation, evaluators have to apply a set of usability heuristics to a system, in order to identify violation of those and in the end assess their severities.

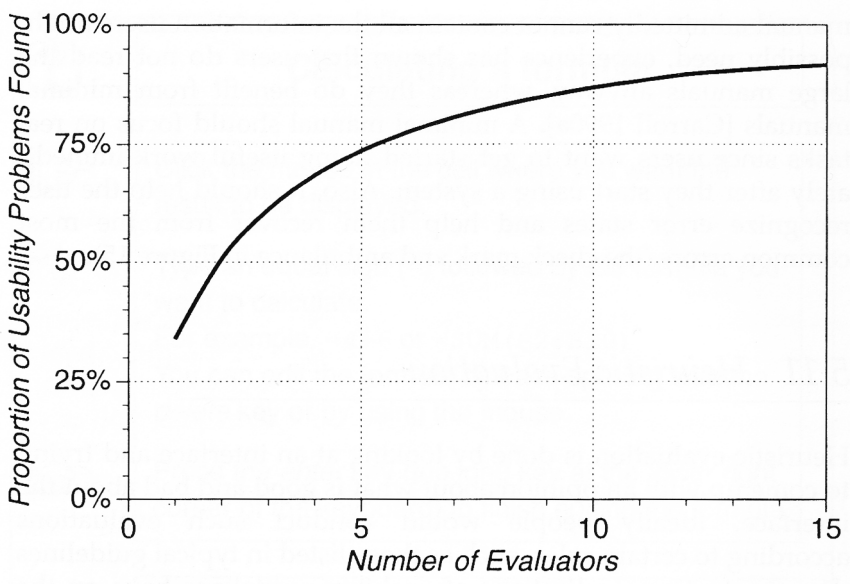


Fig. 2.11: Usability problems found by heuristic evaluation as a function of the number of total evaluators (average results from six studies) [44].

Heuristic evaluation belongs to the branch of usability inspection methods which, in contrast to the empirical methods, are based on the evaluator's informed intuitions about interface design quality [47]. Nielsen described ten heuristics a good interface design should follow [44]:

1. Simple and natural dialog
2. Usage of the users' language
3. Minimization of user memory load
4. Consistency
5. Feedback
6. Clearly marked exits
7. Shortcuts
8. Good Error Messages
9. Prevention of errors
10. Help and Documentation

While the above-listed heuristics represent a solid foundation for an evaluation, they can be altered and extended for the use in a more specific environment [48]. One of the main advantages of heuristic evaluation is that it can be conducted very early in the interface design process since it is applicable to paper and/or electronic mock-ups as well as to already implemented systems. However, a major drawback of heuristic evaluation is that predicted problems may not be congruent with actual problems. This is due to the rather simple implementation of the method and the fact that results are based on the intuition of the evaluators. As a consequence, false problems may be detected on the one hand and real problems may be missed on the other hand. In literature, a maximal usability problem hit rate of 50% per evaluator is described [49]. Nevertheless, based on the subjective nature of this method, different evaluators will most likely tend to find different problems, thus enabling the achievement of better performance through aggregation of all results.

3 Methods

Following established software development methods is considered good practice and helps to streamline the software development process. While this may not be of high importance for individual developers, it is very crucial for projects requiring teamwork. It lies in the responsibility of the developers to adopt the most appropriate method for their task and if required, to modify or even combine it with others from the pool of software development methodologies. Process models illustrate the procedures needed for implementation of those methods. While the early ones were of a linear nature and stipulated sequential processing (e.g. pure Waterfall model [50]), iterative and incremental approaches (derived from the generic Spiral model [51]) were designed for more flexibility in the software development process.

For professional software developers, the advantages of using approved methods for the development process is beyond any doubt. However, the requirement for detailed and accurate specification and documentation, associated with such development methods, is often seen as an obstacle. Not only is it slowing down the innovation process in a project; it also compromises the possibility to react to spontaneous changes of requirements. For this reasons the *Agile Movement* has come into being with the Manifesto for Agile Software Development[20]. The main objectives of agile development are concerned with the emphasis on team relationship, deployment of well tested and working software, close collaboration with clients and the ability to react to changes in requirements during the development process life-cycle [52]. To enforce these efforts, agile development methods (e.g. eXtreme Programming (XP) [53] and Scrum [54]) typically have an iterative structure, propose the work in small groups (6-8 people), refuse detailed documentation and promote a strong involvement of the customer into the development [55]. The popularity of agile methods is mainly due to the rapid growth in the web-based software and mobile application industry, where fast and nimble software development processes are crucial to keep up with client demands and new technologies. However, there are more than a few software development methods, and the selection of the most suitable one for a specific project has to be done carefully since taking wrong decision can easily lead to increased costs or even failure.

Under the conditions of this project, with limited resources in development personnel and a timeframe of approximately six month, the selection of an appropriate software development model was relatively restricted. Slicer extensions are mostly developed in an agile way by the developer community and documentation is preferably provided via

[20] Manifesto for Agile Software Development: http://agilemanifesto.org

the 3D Slicer Wiki pages[21]. The main purpose of these extensions is to support novel research methods, thus implementation requirements are often not completely identified before the coding process. For this kind of software development the agile approach is well suited but also entails the risk of not being able to predict a concrete timepoint for finalization. To avoid this problem, the execution of a pure linear development model in which the required functionality is estimated realistically based on available resources, and not modified during the coding process, is a promising option. However, this procedure aggravates the second goal of this project to create a platform for the further development and prototyping of PET-specific algorithms. For the realization of such a platform, the ability to react to additional feature requests and to incorporate new available Slicer functionalities (if feasible in the context of the work) is of central importance. Hence, the development methods could not be too rigid so that they would prevent any subsequent modifications. In order to cope with these requirements, the software development processes of this work followed a modified Waterfall model with iterative and incremental methods. Furthermore, prototype driven approaches were incorporated for a better understanding of the main requirements.

An illustration of the modified Waterfall model used in this work is shown in Fig. 3.1. In this model, iterative interactions against the process flow are possible only between the design and implementation level and after the evaluation step. This precaution was taken in order to prevent additional specification of requirements once the software design phase started and before a preliminary version of the software was implemented. The possibility to alter or extend requirements is re-enabled after the evaluation of the implementation, which can be used to refine functionality. The implementation itself is based on the incremental build methodology. Functionalities are implemented subsequently offering the possibility to correct errors made in the design phase at relatively low costs since only increments affected by retrospective design changes need to be fixed. In contrast to the software development life cycle, which usually ends with the maintenance process, this project ended with the distribution of a functional extension for Slicer 4.

In the following, the processes performed during the realization of this work, based on the flow of the applied software development method are presented.

[21] 3D Slicer Wiki pages: http://slicer.org/slicerWiki

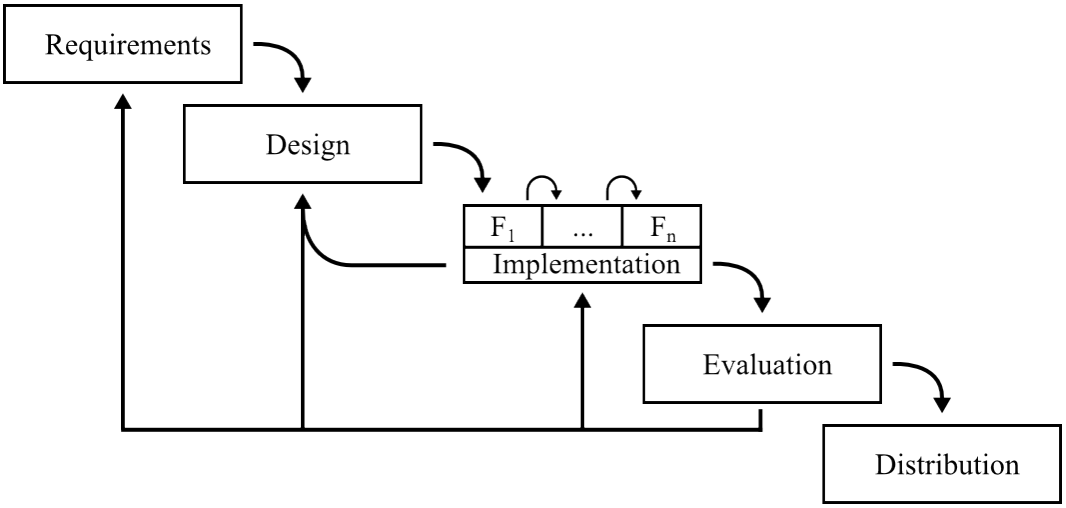


Fig. 3.1: Model of the customized software development method used in this project. Iterative interaction is permitted between design and implementation levels and after the evaluation. Implementation is performed incrementally. Distribution of the final product marks the end of the project.

3.1 Requirements

Requirements engineering is a subfield in software engineering, concerned with the process of formulating and documenting requirements in software development. Software project requirements are often categorized in two groups. First, core functionality requirements include the mandatory properties of the system. Second, ancillary, or non-functional requirement cover optional features that supplement the core functionality [56].

Late incorporation of features that derived from requirements, which were missed at the time of analysis, is one of the main reasons for delays in software development. To reduce these risks, a mock-up driven requirement analysis was conducted in this project. The results from the initial requirements analysis process iteration were used to develop clickable mock-ups of the planned user interface. By using these mock-ups in an additional requirements analysis iteration, it was aimed to improve the yield of requirements gathering and optimize the development process.

3.1.1 Gathering of initial requirements

Chemutury [56] describes the activity of requirement gathering as a combination of elicitation and gathering. Elicitation is the collection of information from users, experts and persons directly concerned with the project. Gathering describes the indirect collection of information from sources like documents and existing applications.

During the initial aggregation of requirements for the software to be developed, the features of two existing Slicer modules were examined. First, the PET/CT Fusion[22] module from 3D Slicer 3 [35], [57] was looked at. As the name suggests, this module is capable of fused visualization of PET and CT image data. Moreover, it provides bodyweight normalized SUV computation for multiple segmented VOIs in one PET/CT dataset. Subsequently, the Change Tracker[23] was examined. A characteristic of this module is its multi-step workflow that guides the operator through a series of parameter input panels before quantification is performed. These workflow steps are as follows. First, two different image data sets, for which changes should be tracked, need to be specified as baseline and follow-up scan. Afterwards, a region of interest (ROI) bounding box is placed around the changing structure of interest. This structure is then segmented in the baseline scan during the third workflow step. A segmentation with the same parameters is later applied to the follow-up scan. The fourth step is concerned with registration and analysis of the segmented volumes from both scans. Therefor, several settings for difference metrics may be made in the input panel. Finally, the volumetric quantification results are presented on the panel of the last workflow step.

In conclusion, examination of both modules revealed that several functionalities might be useful for the implementation of this work. However, none of the modules could be easily extended to realize a tool for longitudinal quantification of tumor changes using PET imaging. While the PET/CT Fusion module has basic PET processing capabilities, it does not support the simultaneous analysis of multiple studies. Whereas the Change Tracker module supports longitudinal analysis but is not designed to function as a tool for visualization and quantification of PET/CT data. As a result, a module that has to be capable of both PET quantification and longitudinal analysis needs to combine the key functionalities from both examined modules.

To extend the field of vision for useful requirements and avoid being too restricted on the available capabilities and methods of Slicer in the early stages of the requirements analysis, the functional principles of longitudinal PET analysis in a commercial system have also been evaluated. By participation in a live demonstration of the PET/CT analysis workflow *syngo.via*[24] (Siemens), important insights about clinical PET analysis were gathered.

---

[22] PET/CT Fusion module: http://slicer.org/slicerWiki/index.php/Modules:PETCTFusionDocumentation-3.6

[23] ChangeTracker module: http://slicer.org/slicerWiki/index.php/Documentation/4.2/Modules/ChangeTracker

[24] syngo.via: http://healthcare.siemens.com/medical-imaging-it/clinical-imaging-applications/syngovia

Table 3.1: Initial requirements classified in core (CF) or ancillary (AF) functionality.

Requirement	Description	Functionality
General requirements concerning the workflow		
R1. Easy to use GUI	Streamlined GUI and workflow optimized for an easy to learn operation.	CF
Requirements derived from PET/CT Fusion module examination		
R2. Fusion imaging	Simultaneous visualization of overlying PET and CT image data using partial transparency of the foreground image and colored lookup tables.	CF
R3. SUV computation	Calculation of Standardized Uptake Values for the segmented volumes of interest.	CF
R4. Registration	Alignment of image data from follow-up PET/CT studies with baseline study.	AF
Requirements derived from ChangeTracker module examination		
R5. Longitudinal Analysis	Import and simultaneous handling of an unrestricted amount of PET/CT studies.	CF
R6. Qualitative Analysis	Display of all segmentation from different studies in one view for qualitative comparison.	CF
Requirements gathered from evaluation of syngo.via functional principles		
R7. Concept of grouping coherent segmentations	Grouping of segmentations of the same structure from different time points.	CF
R8. Easy time point switching	Shuffling between studies (from different time points) should be uncomplicated and accessible at every step of the workflow.	CF
R9. 3D volume rendering	3D volume rendering of PET data for support in detection of tumor hot-spots..	AF
Requirements based on the intention to leverage potentially useful Slicer functionalities		
R10. 3D surface models of segmented VOIs	3D visualization of segmentations in the form of triangulated surface models.	AF
R11. Quantitative analysis visualization	Use of 3D Slicer 4 charting capabilities to visualize quantification results in plots.	CF

3.1.2 Prototyping the Workflow

In software development, mock-ups are often used as rudimentary throwaway prototypes for the GUI and do not provide any other functionality. This strategy is called horizontal prototyping, based on the fact that the whole width of a system is represented while deeper functionality is not implemented yet. Design and generation of the GUI mock-ups for the requirements analysis were realized by using the open-source GUI prototyping tool Pencil[25]. By offering various selections of elements for the easy composition of user interfaces, this tool is superior to standard graphical editors, especially when speaking about productivity. Moreover, the possibility to define user controlled interaction between different mock-ups by exporting projects as HTML pages can help to improve the awareness for the planned procedures.

Inspired by the workflow and GUI setup of the ChangeTracker module, the workflow for the PET quantification module was also divided into multiple steps. The first one represents the import of PET and CT data from a DICOM database. This step has to be completed in order to be able to select PET/CT studies for further analysis. Based on the image data from these studies, detection of lesions and segmentation of those is following in a next step. Finally, quantification is performed based on the segmentation results. As differentiability of the subsequent steps helps to improve user awareness of the current workflow state, a wizard-based design was chosen for the first GUI prototype. In terms of computer science lingo, a wizard is a graphical user interface with subsequent input screens, designed to assist users at certain tasks like software installation routines. Its most distinct feature is the possibility to navigate forth and backwards between the different screens.

3.1.3 Elicitation of Additional Requirements

For the elicitation of additional requirements based on the first GUI mockup prototype, interviews with 3D Slicer developers, clinical researchers and potential future users were planned. For that reason, this project and the mockups of the workflow have been presented at the 2012 NA-MIC Summer Project Week at MIT in Cambridge (MA). A project event, which recorded 88 registered attendees, representing 20 academic sites and 8 companies [58]. NA-MIC project events have been initiated in 2005 and are held twice a year with the main goal to bring together NA-MIC participants and also active and potential collaborators. Maximization of informal interaction between participants and bringing forth the deliverables of NA-MIC is the main goal of these events [59].

Following the initial presentation of this project on the opening day, eight interviews with main developers and potential users were conducted. While the former were

[25] Pencil Project: http://pencil.evolus.vn

looking for a platform to support their efforts in PET data processing with Slicer, the latter were interested in the functionality of a module for longitudinal PET quantification. The summary of additional requirements collected in these interviews is presented in Table 3.2. While requirements formulated by developers mainly concerned data structure and reusability of available code, potential users required improvement of the GUI and inclusion of additional functionality. Further, external developers were interested to use this work as a platform for their own work.

Table 3.2: Additionally elicited requirements classified in core (CF) or ancillary (AF) functionality.

Requirement	Description	Functionality
Additional requirements concerning the workflow		
R12. Organize data in a „Report“	Organization of all data imported from DICOM or generated during the workflow in a data structure called “Report”,.	CF
R13. Organize segmentation results in „Findings“	All segmentations of the same structure from different time points belong to one „Finding“.	CF
R14. Additional study importing	Support of loading of additional studies into an already existing workflow.	AF
R15. Export of results as PDF to DICOM database	Storing of analysis results in DICOM database by creating a PDF file and pushing it to the database.	AF
Additional requirements for the GUI		
R16. Facilitate switching between workflow steps	Based on its sequential paradigm, navigating between non-adjacent steps in a wizard can lead to loss of awareness regarding the workflow state. An alternative apporach to the wizard is required.	CF
R17. Consistent availability of main controls	GUI controls like the slider used for switching studies and the tree view should always maintain the same position in the view and be available in all steps of the workflow.	CF
R18. Minimalize GUI load	Only display the controls absolutely necessary to perform the workflow and by default hide the ones for advanced settings.	AF
Additional functionality requirements		
R19. Integrate Editor module	Integrate Slicer’s manual and semi-automatic segmentation tools into the module.	CF
R20. Multiple volume rendering	Provide multiple 3D viewers for different volume renderings in the comparison view.	AF
R21. HLC chart type for SUVs	Use the High-Low-Close chart type to plot SUV minimum, mean and maximum values.	AF

Based on the results of the second requirements analysis iteration, a new prototype was designed. While the first rudimentary prototype was mainly a composition of clickable mockups, the intention for the second prototype was to streamline the GUI and provide advanced interaction possibilities to improve user feedback. For the implementation of such an improved horizontal prototype, the use of Qt developer tools[26] was considered as appropriate. Given the fact that the GUI of Slicer 4 is based on the Qt framework, a similar look and feel could be realized in the prototype. Using the cross platform Qt Designer for GUI building and the Qt Creator integrated development environment for coding of advanced interactions between GUI elements, a simple application GUI prototype was implemented in a timely manner.

3.1.4 Final Requirements

The third and last iteration of the requirements analysis was conducted based on the second prototype. The application itself and a video demonstrating its use on the basis of an example workflow were distributed to an external nuclear medicine research group. This group has shown interest in using the final application for their work and formulated an additional list of requirements, which are presented in Table 3.3.

Table 3.3: Final requirement additions in the requirements analysis, classified as ancillary functionality (AF).

Requirement	Description	Functionality
Additional functionality requirements		
R22. PET data only visualization	Besides automatic PET and CT fusion imaging, selection for visualization of PET data only should be available.	AF
R23. Support segmentation on CT data	Enabling of the possibility to use segmentation tools on CT data too.	AF
Additional requirements for the GUI		
R24. Simple and informative workflow overview	Display of the current workflow state using a simpler view than the tree view used in both GUI prototypes.	AF

[26] Qt developer tools: http://qt.digia.com/Product/Developer-Tools/

3.2 Design

The successive design of a software system is a procedure that can be approached from four different directions. *Top-down*, *bottom-up*, *inside-out* and *outside-in* are the catchphrases used to describe these directions. The spatial perception of this model is shown in Fig. 3.2. Ludewig and Lichter [55] explain the different approaches as follows.

Starting to design software based on an abstract task or problem with the aim to develop a detailed and concrete solution is considered to be a *top-down* procedure. Recursive partitioning of the problem until an elementary level is reached (commands of the programming language) is the most common strategy in this kind of development. In contrast to this analytical approach, *bottom-up* development is based on repeated synthesis of low-level components until the final solution is obtained. Beginning the development of the software with the internals of a system and working towards a user interface can be described best as an *inside-out* approach. Typically this is the case when additional functionality is added to an already existing system, e.g. a graphical user interface for more convenient operation of an existing command line application. *Outside-in* is the last of the four different directions, implying the design of a system based on an already existent interface. Use of the *outside-in* approach is most appropriate in cases where stakeholders have evaluated a horizontal prototype and details of implementation are not specified yet.

Due to the results achieved in the requirements analysis of this project, the detailed second GUI prototype was used for further design, following the *outside-in* strategy. In the remainder of this section, the architectural design for the software is presented. Special focus is given to the modeling of required class hierarchies, event- based GUI behavior and optimized handling of user interaction.

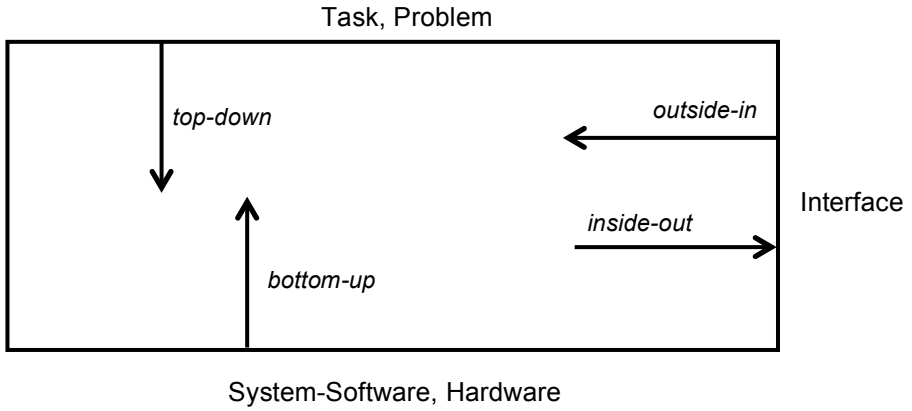


Fig. 3.2: Directions for development in software design [55].

3.2.1 Module Type Selection

Slicer supports three different types of modules [60]. Before the implementation of a new module, developers have to make the decision whether a command line interface (CLI), a loadable module or a scripted module type is most suitable for their tasks.

CLI modules are standalone executables or shared libraries, providing only limited input/output parameter complexity. Moreover, they do not support user interaction other than selection of input and output parameters in a GUI, generated from an XML (Extensible Markup Language) schema file. Autonomically working algorithms, once input parameters are passed, are best implemented and distributed as CLI modules.

Loadable modules in contrast, offer full control over their own GUIs, access to all internal Slicer data structures (MRML data, program logics and display managers) and to the APIs of the toolkits used within Slicer. These modules are distributed as shared libraries, which are loaded at startup of the application. Complex user-system interaction, data processing and visualization can be realized within a loadable module. However, the increase in possibilities and functionalities depends on a more extensive implementation following the MVC design pattern of the main application.

While CLI and loadable modules require implementation in C++, scripted modules in Slicer 4 are written using the programming language Python. Similar to the loadable ones, scripted modules offer access to Slicer internals and to the APIs of the toolkits that are fully wrapped for Python use (VTK, ITK, MRML, CTK, Qt). With access to CTK and Qt, scripted modules also provide full control over their GUI. The main advantage of implementing a scripted module lies in the increased productivity based on the simplified programming realized by the scripting language. While C++ written code has to be recompiled after every change, scripted Slicer modules offer a mechanism capable of updating the module during execution. Despite of all its advantages, the generation and integration of additional MRML data types is not possible within a scripted module since the MRML library is written in C++.

Requirements analysis has shown that using only data types from the MRML library will not suffice the needs for modeling of the data structure for the new module. Taking this into account, the implementation of the work as a loadable module is required. However, the advanced build mechanism of Slicer offers the possibility to integrate scripted classes into loadable modules in order to create a hybrid module written in C++ and Python. For the creation of such module, a majority of components can be implemented using Python scripts, apart from the mentioned custom MRML classes, and the logic responsible for integration of those into the main application. Given that possibility, the decision was made to implement this work in the form of a hybrid module. In so doing, the possibility to integrate own MRML data types while still being able to leverage the advantages in productivity using the supported scripting techniques was retained.

In the following section the design for module specific MRML classes is presented and an overview of the event-driven interaction of GUI elements with the module's Model and Controller classes is given.

Please note that although the software tool implemented in this work will be distributed in the end as an *extension* for Slicer, it will be mostly referred to as *module* in the remainder of this work. This is due to the fact that the implementation of an extension does not differ from the one of a module. The only difference is the fact that extensions do not belong to the standard distribution of Slicer but can be installed on demand (for further details see section 2.4.5).

3.2.2 Model Classes - MRML

In the MVC driven architecture of Slicer, the Model is based on classes and hierarchies defined in the Medical Reality Markup Language and accessible via the MRML library. When implementing own classes derived from the base class `MRMLNode`, Slicer's event driven pipeline functionalities are automatically inherited and seamless integration into Slicer's `MRMLScene` is possible. The abstract superclass for all specific types of MRML classes is the `vtkMRMLNode`. As the nomenclature suggests, this class inherits a VTK class, namely `vtkObject`[27]. This abstract VTK base class mostly provides methods for the implementation of an event-driven architecture offering mechanisms for registration of observing instances and invoking of events. The `vtkMRMLNode` class extends this functionality with variables and methods for identification and organization in the `MRMLScene`, duplication, serialization and deserialization.

All classes in the Model of the module to be implemented derive directly from `vtkMRMLNode` and overwrite the inherited abstract methods for duplication, serialization, deserialization and console output. In order to realize the data hierarchy that was elaborated using the prototypes in the requirements analysis, four new classes needed to be implemented for the Model. Following Slicer naming guidelines, these classes are named as follows:

- **`vtkMRMLLongitudinalPETCTReportNode`** (further referred to as `Report`): Organizes patient information and manages the different Studies and Findings of the analysis workflow.

- **`vktMRMLLongitudinalPETCTStudyNode`** (further referred to as `Study`): Handles PET and CT image data loaded from a DICOM study.

[27] Documentation of `vtkObject` classs: http://vtk.org/doc/nightly/html/classvtkObject.html

- **`vtkMRMLLonigtudinalPETCTFindingNode`** (further referred to as `Finding`): Represents a structure of interest that can be segmented for quantification across the image data of all studies.

- **`vtkMRMLLongitudinalPETCTSegmentationNode`** (further referred to as `Segmentation`): Stores the quantification result that were computed for a certain segmentation in the image data of a specific study.

A simplified UML (Unified Modeling Language) class diagram of the above listed classes is shown in Fig. 3.3 (see appendix A for detailed UML diagrams). An object of `Report` represents the backbone of every PET/CT analysis workflow. It is bound to a single patient and contains image data from his PET/CT DICOM studies and the `Findings` that are created during analysis. All data loaded or generated for an analysis workflow can be accessed through the respective `Report` object. Objects of `Study` are concerned with PET/CT imaging data. They may each contain one PET and one CT image series from a DICOM PET/CT study. Moreover, `Study` also provides additional infrastructure required for segmentation, 3D volume rendering and registration of its PET/CT image data. While `Finding` is used to represent a tumor or a reference structure that is present in the PET image series of at least one PET/CT study, `Segmentation` handles an actual segmentation for such a `Finding` and stores the quantification results. In every `Finding`, one `Segmentation` may be created for each `Study` available in `Report`.

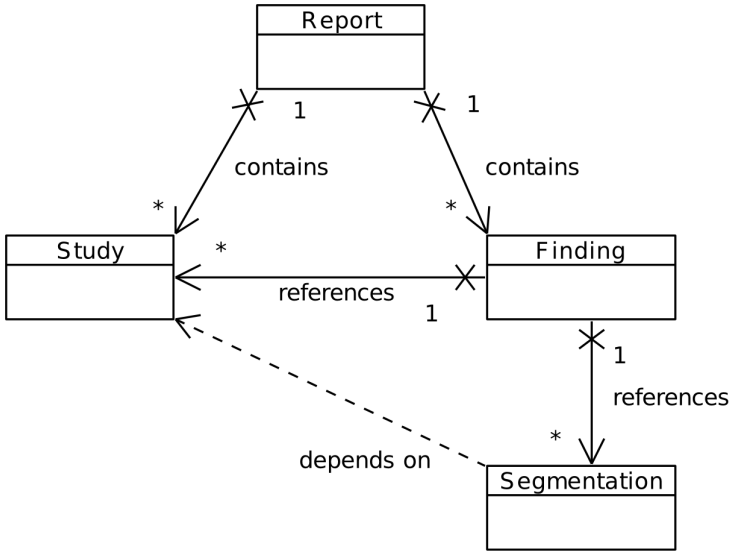


Fig. 3.3: UML class diagram of the Model classes (MRML)

3.2.3 View Classes - Widgets

All GUI widgets implemented in this work are written in C++, although the possibility to use Python scripting for the composition of the module's GUI is given. In so doing, the individual widgets are directly accessible as Qt conform objects and are not encapsulated within Python script files. By separating the GUI in multiple widgets containing coherent controls, a higher modularity can be achieved. Creation of customized Qt signals is also a great benefit when coding Qt widgets in C++. This allows bundling of different signals into a single one specialized for the specific task. Not only does this reduce the logical complexity of the application, it also helps to keep the code clean. Another functionality provided by VTK and not yet accessible from Python is the connection of VTK based events with slots in Qt objects. Given the fact that all classes from the MRML-Library derive from `vtkObject`, this opens the possibility for implementation of autonomically updating GUI widgets. Being able to receive a signal whenever a MRML object has been modified, a widget containing specific information about this object can automatically react to the modification and update itself without any interaction from the application's logic.

Designing ajar to the GUI prototype developed for requirements analysis, it was advisable to break down the module's default GUI into multiple widgets. For every single workflow step a separate widget was implemented. In addition to these, a widget for facilitated review of the current workflow state and a widget for modification of workflow related settings was provided (see Fig. 3.4).

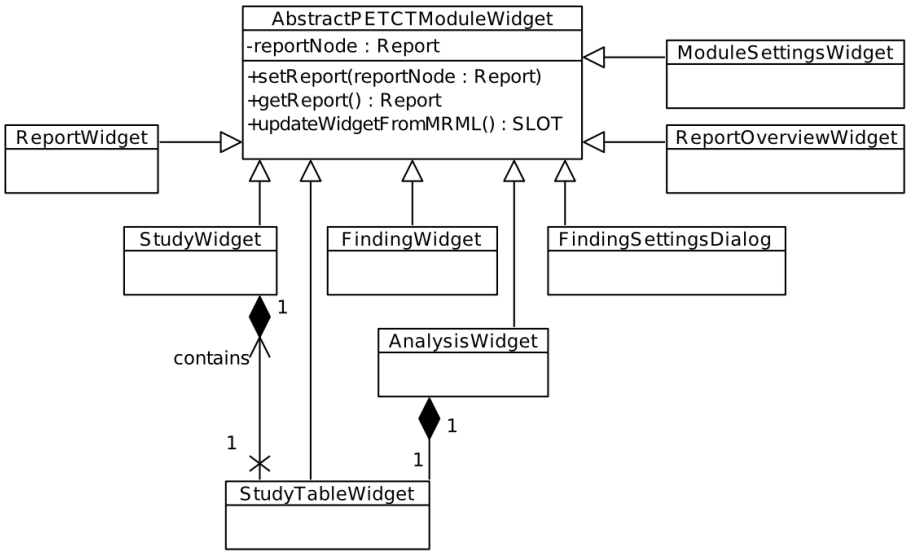


Fig. 3.4: UML class diagram of the View classes. All GUI widgets derive from the *`AbstractPETCTModuleWidget`*. A pointer to the active `Report` is stored in each widget, thus enabling autonomic updating of the widget whenever this `Report` object is modified.

The abstract class *AbstractPETCTModuleWidget* represents the base class for all widgets of the module. Its only attribute is a pointer to a Report object. Whenever a Report is assigned to an object of the class, the object's specific VTK-event for modification is automatically connected to the abstract *updateWidgetFromMRML()* slot method using the VTK event - Qt slot connection functionality. All subclasses are required to overwrite this method in order to update their visual representation upon modification of the connected Report object. The ReportWidget represents the GUI widget for the first workflow step. Selection between all Report nodes available in the active MRMLScene and information about the respective patient are provided in this widget. Using the functionalities of its built in StudyTableWidget, the StudyWidget offers the possibility to add or remove Study objects to or from the analysis workflow, which is done in the second workflow step. The FindingWidget provides options for creation, selection or deletion of Finding objects and the possibility for localization of those in the image data. After creation of a Finding object, the FindingSettingsDialog is used to specify the name and color identifier of the newly created Finding. Furhtermore, a widget representation of Slicer's Editor module is also integrated in this widget. The built-in Editor module provides direct access to the tools required for segmentation. Handling of Finding and Segmentation objects is part of the third workflow step. Similar to the StudyWidget, the AnalysisWidget contains a built in StudySelectionTableWidget for individual comparison of analysis results in the forth workflow step. Selection between qualitative or quantitative analysis is also possible from within this widget as well as the export of analysis results to disk.

Insights gained from prototype development suggest that only one workflow step related widget should be visible at a time. Since these widgets are all individual instances at runtime, this requirement is easily implementable. However, for an even better awareness of the PET/CT analysis process, the ReportOverviewWidget provides a facilitated overview of the current workflow state. Moreover, it can also be used as a browser for quick access to image data from specific PET/CT studies or segmentation results of certain Findings.

As already mentioned above, autonomic updates of GUI widgets are handled by the connections of the abstract base class's Report object to the overwritten updateWidgetFromMRML() slots. Outgoing communication is realized by signal emission. Each time a user interacts with one of the GUI controls, the widget containing this element emits a specific signal, which then can be processed in the module Logic.

3.2.4 Controller Classes - Logic

While the classes of Model and View are written in C++, almost all classes of the module's Controller (Logic) are written in Python. The sole exception is a single C++ class required for registration of the module's MRML nodes to the main application. A total of four Python classes represent the module's Logic (see Fig. 3.5).

The `DICOMPETCTPlugin` is a class exclusively dedicated to the import of valid PET/CT studies from Slicer's DICOM database. In Slicer 4, this is realized using the DICOM module's plugin functionality. Objects of classes that inherit Slicer's `DICOMPlugin`[28] class and implement its virtual methods are automatically registered to the DICOM module during application startup.

As explained in the previous paragraph, user interaction in the module's GUI is forwarded to the application Logic using `signal` emission. During composition of the GUI widgets, these `signals` are connected to the corresponding `slots` by an instance of the main Logic class (`LongitudinalPETCTModuleLogic`). Besides access to all GUI widgets of the module, the class also is able to store user selected `Report`, `Study` and `Finding` objects. Having access to this data, the required reaction to user interaction can be easily implemented in the respective `slots` of the main Logic class.

Non-`slot` methods of `LongitudinalPETCTModuleLogic` are either concerned with the appropriate visualization of image data for a specific workflow step or support of the segmentation process. Two additional Controller classes grant support for such main-application related operations. On the one hand, the `VisualizationHelper` class contains methods for switching between different visualization modes and provides access to Slicer's objects for data visualization such as the 2D Slice Viewers and the 3D Volume Rendering Viewers or the Chart View representation. On the other hand, the `SegmentationHelper` class mostly provides methods for processing of segmentation results in the context of the module.

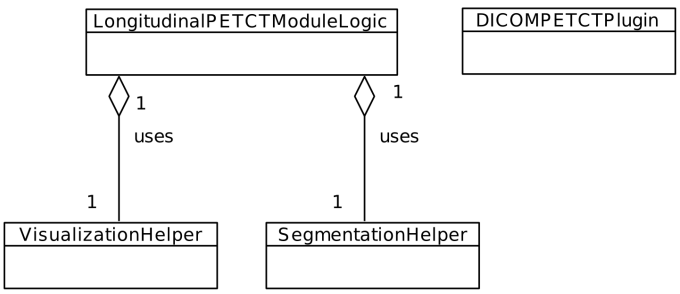


Fig. 3.5: UML class diagram of the Controller classes.

[28] DICOM Plugin: Class Reference:
http://slicer.org/doc/html/classDICOMLib_1_1DICOMPlugin_1_1DICOMPlugin.html

3.3 Implementation

Based on the underlying software development process model of this work, the implementation of the module was conducted by following an incremental build approach. In contrast to the horizontal prototyping, done in the requirements analysis, the incremental implementation represents an evolutionary vertical prototyping process. Fully functional subparts of the software are successively coded, tested and reviewed before their integration into the system.

Implementation of the software realized in this work was partitioned into eight build increments. Each increment depends on the functionality of the previous one and consists of certain parts from the Model (MRML), View (Widget) or Controller (Logic) of the module. The process of implementation was oriented to the GUI widgets representing the different workflow steps of the PET/CT analysis. However, before such a GUI widget could be implemented, the data structure required for full functionality of the widget had to be made available. Then, after the widget was finalized and offered all designated functionality, the module's Logic was enhanced to ensure seamless integration of the newly implemented elements and full support for user-system interaction. In the Slicer main application GUI (see Fig. 3.6), the module's GUI was integrated into the Module Panel (on the left).

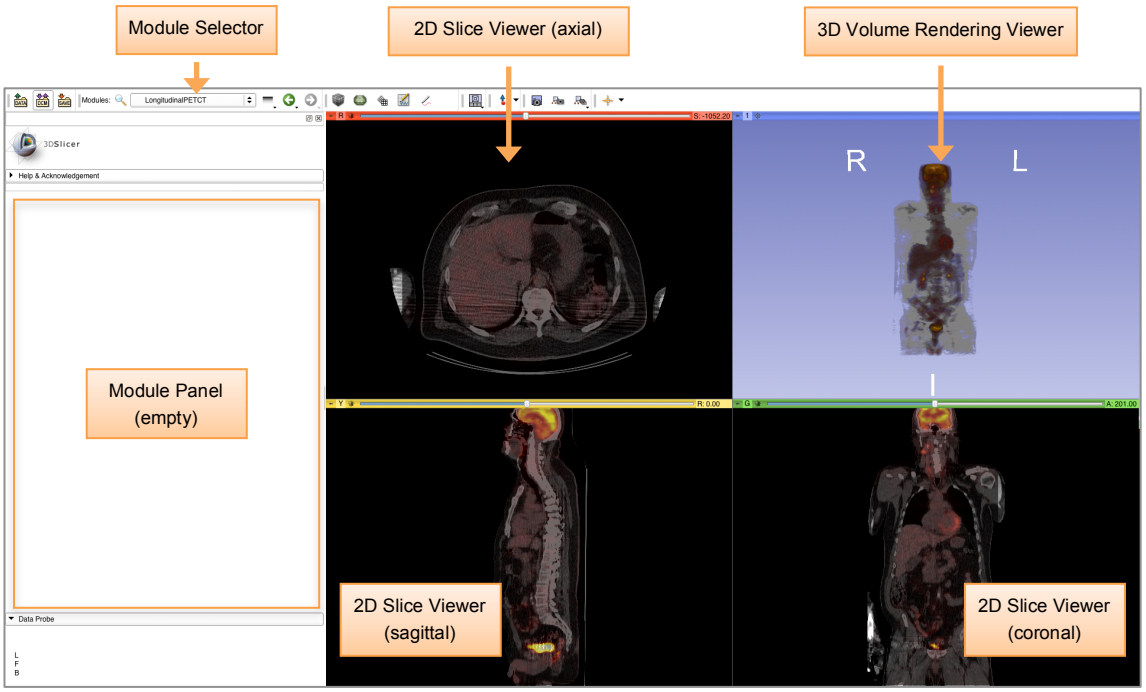


Fig. 3.6: Slicer main application GUI with empty Module Panel and PET/CT Fusion visualization of loaded image volumes. The new PET/CT analysis module is integrated into the Module Panel (left) and can be selected with the Module Selector (top-left). During an analysis workflow, extensive use of the 2D Slice Viewers and 3D Volume Rendering viewer is made for visualization of 3D image data and segmentations results.

In the first build increment, scanning of Slicer's DICOM database for patients with valid PET/CT studies and loading of PET and CT image volumes from these studies into the `MRMLScene` was realized. Provision of access to the loaded data belonged to the implemented functionality of the second increment. Selection of image volumes for further segmentation and analysis together with advanced 2D and 3D visualization were objectives for the third increment. Afterwards, the foundation for organization of segmentation and quantification results was laid in the forth increment by implementation of the `Finding` data structure. This one was introduced in the second GUI prototype. Actual segmentation and quantification of PET image data was part of the fifth increments' functionality implementation along with 2D and 3D visualization of the segmentation results. The sixth increment was concerned with provision of overview of the workflow and quick access to key data. Qualitative and quantitative analysis visualization, as well as the possibility to export the analysis results to disk, were implemented in the seventh increment. Finally, saving and loading of the generated data and the possibility to modify PET/CT related visualization and computation settings in the module was realized in the last build increment.

In the following, the techniques deployed in the incremental building of the above-mentioned module subparts are presented. The module implemented in this work is further referred to as Longitudinal PET/CT Analysis module.

3.3.1 Increment 1 - Loading of PET/CT Data from DICOM Database

Using only built-in Slicer DICOM plugins to import PET/CT studies for a longitudinal multi-study analysis implies a cumbersome manual selection of the related PET and CT image series. Hence, the implementation of a new plugin that is capable of automatic selection and organization of the appropriate PET and CT DICOM data was advisable. Such a plugin needs to summarize and present all coherent image data in one selectable element.

The objective for the first build increment was to implement a new DICOM plugin specialized on loading of PET/CT studies from DICOM database into the `MRMLScene` and to provide the data structure required for this transition.

3.3.1.1 MRML Implementation

In the Longitudinal PET/CT analysis module, an instance of `Report` represents the backbone of every PET analysis workflow. In the first build increment, a `Report's` main objective is to provide access to the image volumes loaded from the DICOM database. Objects of the `Study` class handle organization of these volumes. At the beginning, a `Study` object is implemented to store the unique identifiers of each, a PET

and a CT image volume, and provide methods for assignment and access of those. In the implementation of `Report`, `Study` objects are maintained in a vector[29] variable, in which they are sorted chronologically according to the acquisition date of their image data.

All Model classes implemented in this work derive from the MRML-Library base class `vtkMRMLNode`. In doing so, compatibility with Slicer's `MRMLScene` and access to the advanced event-handling functionality of Slicer is provided automatically.

3.3.1.2 GUI Widget Implementation

The required interface for loading of image data from Slicer's DICOM database into the `MRMLScene` is already provided by the DICOM-Browser from the DICOM module. Hence, there was no need for implementation of a new GUI widget in the context of this build increment.

3.3.1.3 Logic Implementation

The scripted Logic class `DICOMLongitudinalPETCTPlugin` represents the DICOM plugin provided by the Longitudinal PET/CT module. It belongs to the module Logic, since it only realizes the actual loading of DICOM data and does not provide an own GUI. An interface for access of this plugin is supplied by Slicer's DICOM-Browser.

To ensure full compliance with the DICOM plugin infrastructure, the plugin must inherit the base class `DICOMPlugin` and overwrite two of its virtual functions. One of these is required for the examination of the user selected DICOM data hierarchy while the other handles the transition of DICOM data into derivates of `vtkMRMLVolumeNode` objects.

In the `DICOMLongitudinalPETCTPlugin,` the overwriting implementation of the former function is dedicated to parsing of user-selected DICOM data hierarchies for studies with PET and CT image series. Only studies that contain image series from both types are considered valid PET/CT studies and qualify for the longitudinal PET analysis workflow. References for access of the image data from those PET/CT studies are cached during the examination process. The overall examination results are summarized into a single data object, which is returned to the DICOM module. There, it is presented as a single entry in the DICOM-Browser amongst the examination results of the other DICOM-Plugins.

Loading of PET/CT DICOM image data into Slicer's `MRMLScene` is implemented in the second overwriting function. Access to image data that has been cached during the examination step can be accessed using the summarized result object. Similar to the

[29] Sequence container implementing a data array capable of automatic resizing and random accessing

approach in the default DICOM-Plugin for loading of scalar volumes, all selected PET and CT image series are transformed into `vtkMRMLScalarVolumeNode` objects. In addition to that, related PET and CT volumes are each assigned to newly generated instances of `Study`, where they are stored together with PET/CT study and image series parameters, which is important for PET quantification. Finally, all unique identifiers of the `Study` objects and information about the patient are stored in an instance of `Report`, which is then added to the `MRMLScene`.

An overview of the above presented process sequences is given in Fig. 3.7.

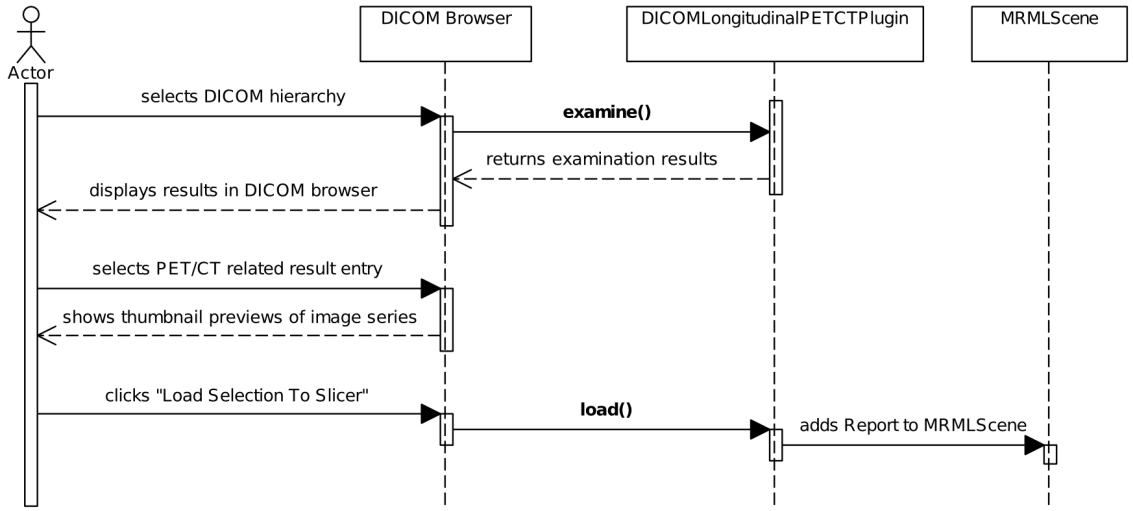


Fig. 3.7: UML Sequence Diagram for loading of PET/CT DICOM studies into the MRMLScene using the DICOM-Browser and the DICOMLongitudinalPETCTPlugin.

3.3.2 Increment 2 – Selection of Active Report Object

With the finalization of the `DICOMLongitudinalPETCTPlugin` in the first build increment the requirements were met for the implementation of the actual module in the second build increment. The main goal of this increment was to enable individual selection of the `Report` objects that are loaded into the `MRMLScene`. By design, only one `Report` object can be active in the module at a time, thus only data belonging to this `Report` is editable in the module. In the following, the implementation of a GUI widget for determination of the active `Report` and the implementation of application logic required to support such selection is presented.

3.3.2.1 GUI Widget Implementation

The starting point of every longitudinal PET/CT analysis workflow is initiated by the selection of a `Report` object. Hence, a GUI, which provides the controls for such selection, is required. The appearance of this interface is shown in Fig. 3.8.

For the actual selection, a combobox[30] presenting all available `Report` objects from the `MRMLScene` is used. Patient information, stored in the selected instance of `Report`, is also presented in this GUI widget. If desired, this information can be hidden using the button located at the bottom. In case that these specifics are not of interest or when only low screen resolution is available, this feature helps to improve the visual appearance of the widget by providing a tidy and compact interface.

An information label on the top right, which appears in this and all further implemented GUI widgets of the module, represents a novelty approach in Slicer GUI design. Hovering the mouse cursor over this label triggers the temporary display of a textbox with information about the widget's functionalities.

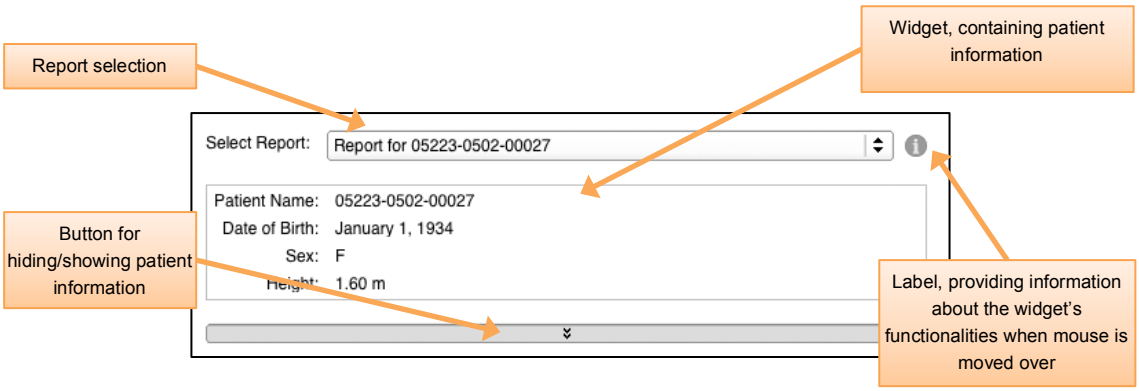


Fig. 3.8: GUI widget for active `Report` selection

3.3.2.2 Logic Implementation

Implementation of the Logic for active `Report` selection contains of a single `slot` method and a variable representing this current active `Report`. The method is able to receive `signals` that are emitted from the GUI whenever the `Report` selection changes. Then, the selected Report is assigned to the previously mentioned variable (see Fig. 3.9).

[30] GUI control, represented by a combination of a drop-down list and a textbox.

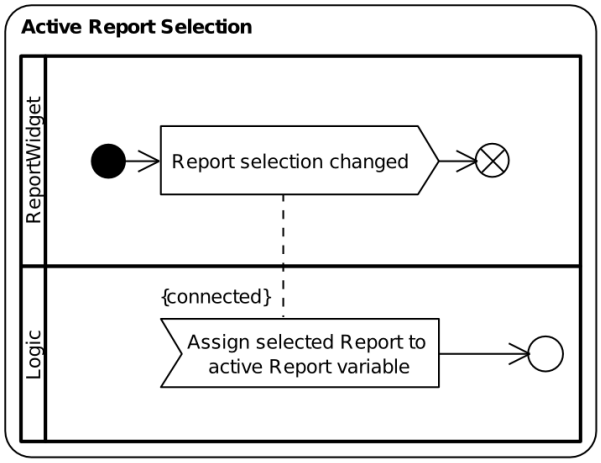


Fig. 3.9: UML Activity Diagram for the selection of the active `Report` object in the first workflow step.

3.3.3 Increment 3 – Visualization of Image Data

By design, PET and CT image volumes referenced by the loaded `Study` objects are not by default unlocked for segmentation in the module. In order to unlock them and by this means add them to the further analysis workflow, user interaction is required.

Providing an overview of the selected `Report's Study` objects and the possibility to select or deselect those for segmentation and further analysis was the objective of the third build increment. The following three paragraphs describe how the module implementation was enhanced to provide the required functionality for such presentation and selection of `Study` objects.

3.3.3.1 MRML Implementation

Based on the implementation from the first increment, `Study` objects only provide the functionality to manage PET and CT image volumes. In order to support the selection for segmentation and provide all required data for 2D and 3D visualization of the image volumes, enhancement of the `Study` class was required. Besides a boolean[31] variable for labeling the `Study` object as selected for segmentation, three additional variables were included. Since segmentations in Slicer are performed on `vtkMRMLScalarVolumeNode`-based objects called label map volumes, `Study` objects require a variable to store the unique identifier of such volumes. For 3D volume rendering visualization, objects of the type `vtkMRMLVolumeRenderingDisplayNode` are used. On that account, the Study class

[31] Data type limited to two possible values, usually denoted as `true` and `false`.

was enhanced to be able to store unique identifiers of such objects too. The last of the new variables is used to keep track of `vtkMRMLLinearTransformNode` objects. These can store the parameters of a linear transform, which is used for registration of one `Study` object's image data with another's. Naturally, methods for access and modification of the listed variables were also added to the implementation of the `Study` class. In contrast to the rather extensive adaption of the `Study` class, enhancement in the `Report` class was confined to an additional variable, and it's access and modification methods. This variable is responsible for storing the `Study` object, which has been selected by the user in the GUI to be the current active `Study`. Ajar to the concept of active `Report`, only the image volumes of one `Study` can be visualized by the module at a time, this being the one selected as active.

3.3.3.2 GUI Widget Implementation

The GUI widget implemented in the third increment offers twofold selection possibilities. Firstly, it can be used to select `Study` objects in order to unlock them for segmentation in the following workflow steps. Secondly, it allows the selection of the active `Study` whose PET and CT image data can then be visualized in the 2D Slice and 3D Volume Rendering viewers of the main application. This GUI widget is presented in Fig. 3.10. A tabular presentation of all `Study` objects available in the active `Report` is the main component of the widget. Corresponding to the acquisition of their image data, the Study objects are listed chronologically from top to bottom. Information, which has been stored in the `Study` objects during the loading process from the DICOM database, is presented in the numerous columns of the table. Details of DICOM specific identifiers, acquisition dates and information used for the computation of SUVs are shown. By checking the checkboxes in the first column of the table, `Study` objects can be selected for further segmentation and analysis. Marking a row in the table, by clicking on one of its cells, can be used to select the active `Study` object. However, only rows in which the checkbox in the first column is checked can be marked. The reason for this restriction is that only the image volumes of `Study` objects, which have been added to the workflow (marked for segmentation), should be visualized. Likewise to the `Report` selection GUI widget implemented in the previous increment, an information label is available in the top right corner, explaining the widget's features and their usage.

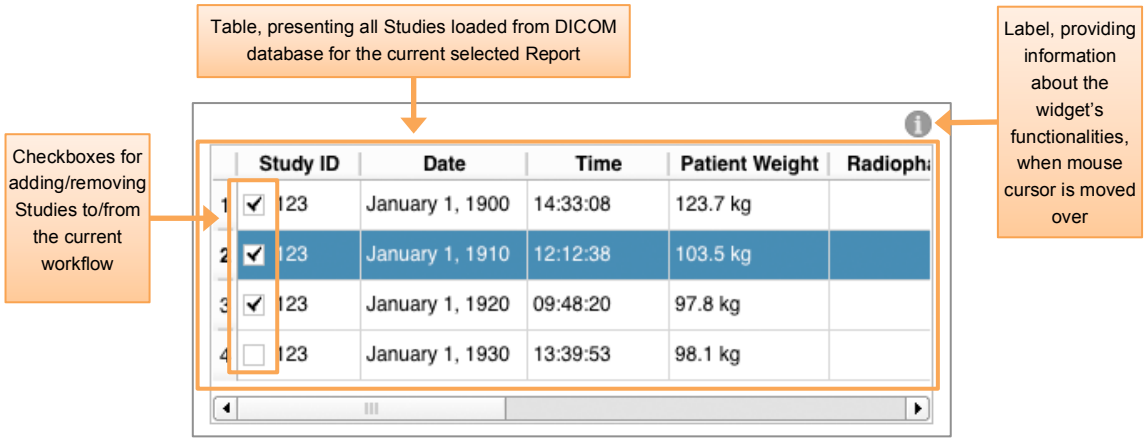


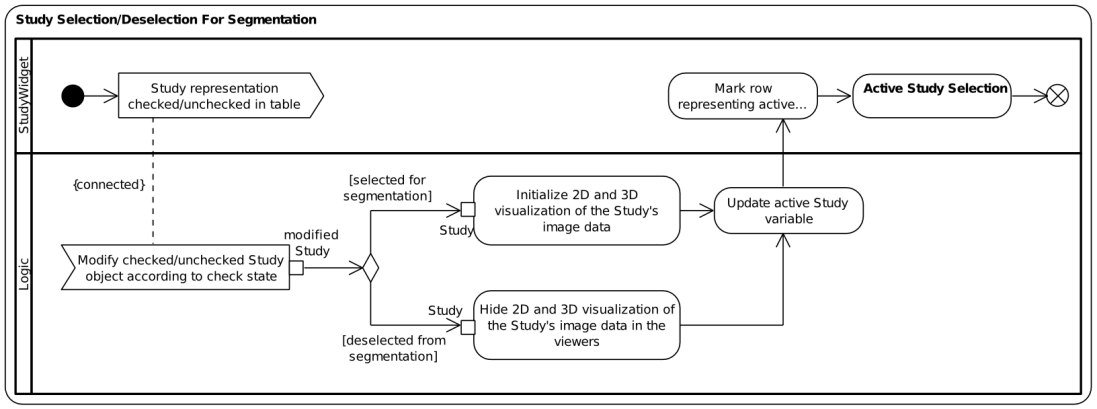
Fig. 3.10: GUI Widget providing `Study` objects overview and selection possibilities.

3.3.3.3 Logic Implementation

For the integration of the new GUI widget and leverage of the enhanced MRML data structure, the implementation of two `slot` methods was required in the module's Logic class. While one `slot` handles the selection and deselection of a `Study` object for segmentation, the other one is responsible for assigning the active `Study` to the corresponding variable. A UML activity diagram of the process flows supported by the Logic enhancement in the third increment is shown in Fig. 3.11. When a `Study` is selected for further segmentation, the respective `signal` emitted by the GUI is received by the Logic and the `Study` object's variable for this selection status is set to `true`. When deselected, this variable is set to `false`. If selected for segmentation, the Logic initializes the visualization of the `Study` object's image data. This process includes the generation of a `vtkMRMLScalarVolumeNode` label map, a `vtkMRMLVolumeRenderingDisplayNode` for volume rendering visualization and a `vtkMRMLLinearTransformNode` object required for registration of the image volumes. After creation, the unique identifiers of those objects are all stored in the respective `Study` object. Subsequently, the variable holding the active `Study` is updated. A `Study` selected for segmentation is automatically set as the active `Study` and the Logic manages the visualization in the 2D Slicer and 3D Volume Rendering viewers. On the other hand, a `Study` deselected from segmentation has all its image data removed from visualization and if available, an adjacent selected `Study` is set as active by the Logic (Fig. 3.11a).

The second `slot` added to the Logic in this increment reacts to the `signals` emitted by the GUI when the active `Study` is changed. After reassignment of the variable for active `Study`, the method updates the 2D Slice Viewers and the 3D Volume Rendering viewer with the image data of the new active `Study` (Fig. 3.11b).

a.



b.

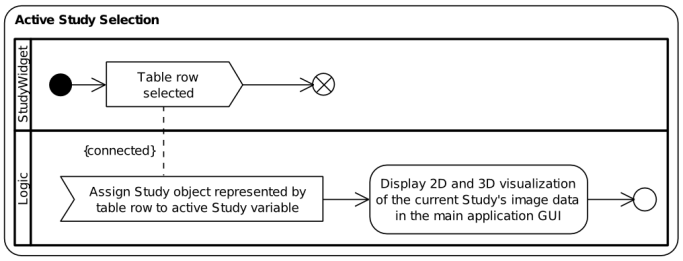


Fig. 3.11: UML activity diagrams for selection of `Study` objects for segmentation (a) and as active Study (b).

3.3.4 Increment 4 - Creation and Selection of Findings

Assessment of tumor changes in time series PET requires the segmentation of the tumor in each single image series in order to be able to analyze the changes. The Longitudinal PET/CT module is restricting neither the amount of time-displaced PET/CT studies that can be examined nor the amount of different structures, which can be segmented in each PET image series. For that reason, the `Finding` data structure was introduced during the requirements analysis and design phase of this project. The goal of the forth build increment was to implement a new MRML class for this data structure and provide a GUI widget for the creation and selection of it.

3.3.4.1 MRML Implementation

At the beginning, the new MRML class `vtkMRMLLongitudinalPETCTFindingNode` (further referred to as `Finding`) provided and handled four different variables. Since distinguishability of `Finding` objects is important for the overview in the analysis workflow, a unique color can be assigned to each `Finding` object after creation. Such color derives from a built-in Slicer color table, which provides the default colors for segmentation of the Editor module. However, only the identifier needed to access the color from the table is stored in a `Finding` object. In the workflow, placing an ROI bounding-box in the 2D Slicer Viewers specifies the location of the Finding in the image data. The coordinates of the ROI center and its radius are stored in two different

three-data array variables. The last variable is a `boolean` flag which indicates whether an ROI for the `Finding` object has been placed by the user.

For full integration of the new `Finding` class on the MRML level, enhancement of the `Report` class was required too. Similar to the vector-based management of `Study` objects, `Finding` objects are also maintained in a `vector` variable in `Report`. An additional variable was also included for the purpose of storing the user-selected active `Finding`.

3.3.4.2 GUI Widget Implementation

Using the Slicer built-in combobox GUI widget for `MRMLNode` selection (`qSlicerMRMLNodeComboBox`) allowed compressing of the controls needed for creation, selection and deletion of `Finding` objects into one control in the user interface. The GUI implementation in this build increment consisted of a widget (see Fig. 3.12.a) and a dialog (see Fig. 3.12.b). In the widget, the above-mentioned combobox is located in the middle of the top area of the widget. When expanded, it exposes access to the controls for creation of new and deletion of existing and selected `Finding` objects. Underneath located is the button for ROI placing, which is activated subsequently to the selection of a `Finding` object. Clicking this button activates the module's functionality for placing an ROI bounding box in the 2D Slice Viewers of the main application. Visualization of such bounding boxes in the viewers can be activated or deactivated using the control for ROI visibility. Following the design of the GUI widgets implemented in previous increments, an information label is available in the top right corner.

In the dialog (see Fig. 3.12.b), a preset of 5 different `Finding` types is given. These consist of tumor, lymph node, liver, cerebellum and aorta, and represent structures that are often segmented in PET analysis. Thus, quick setting of `Finding` parameters using these presets can improve the usability of the workflow. In case that none of the presets matches the `Finding` intended to create, manual setting of the name and selection of a color identifier is possible using the GUI controls located underneath the presets buttons.

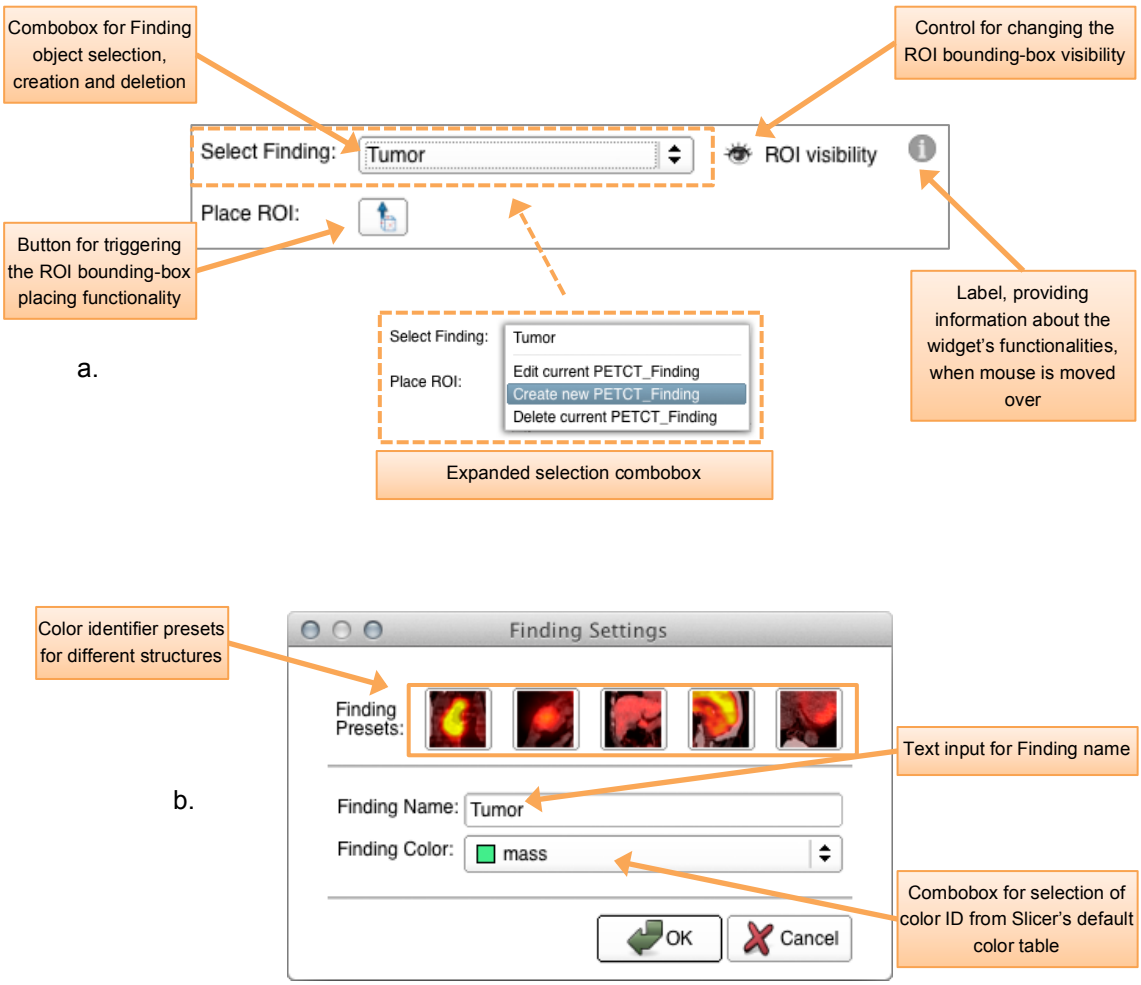


Fig. 3.12: GUI Widget for creation and selection of `Finding` objects (a) and Dialog for setting of tumor name and color identifier (b).

3.3.4.3 Logic Implementation

While `Report` and `Study` objects are generated automatically by the application, the existence of `Finding` objects is fully controlled by the user. Instantiation of said objects and their integration into and removal from the `MRMLScene` is performed automatically by the `qSlicerMRMLNodeComboBox`, which was integrated into the module's user interface in the previously presented GUI widget. In those circumstances, the enhancement of the module's Logic was focused on the further processing and customization of the `Finding` objects for the analysis workflow. Once a new `Finding` has been added to the `MRMLScene`, the Logic receives the object in a `signal` from the GUI and adds it to the `Report` as the current active `Finding` (Fig. 3.13.a). The method for this assignment is also a `slot`, which is used to receive the `signals` from the GUI whenever the active `Finding` is changed (Fig. 3.13.b). In case that a Finding has been created in the widget, the dialog pops up and gives the user the possibility to

specify the name and color identifier of the created object. A `signal` containing these parameters is emitted by the dialog and processed by the Logic once the dialog is closed.

When a `Finding` object is deleted using the combobox, the GUI widget emits a `signal`. This `signal` is then processed by the Logic and leads to the removal of the `Finding` from the active `Report` before it gets completely removed from the `MRMLScene` (Fig. 3.13.c). Provided that a `Finding` object has been successfully selected in the GUI, the user can trigger an ROI bounding box placing functionality. The interactive placing is executed with two subsequent clicks in one of Slicer's 2D Slice Viewers. After a first click specifies the center of the ROI, a second click defines its extent. Recording and translating the position of those two clicks from the 2D Slice Viewers into the 3D image data coordinate system is also performed by the module Logic. Eventually, the resulting coordinates and radius values of the ROI are stored in the active `Finding` object (Fig. 3.13.d).

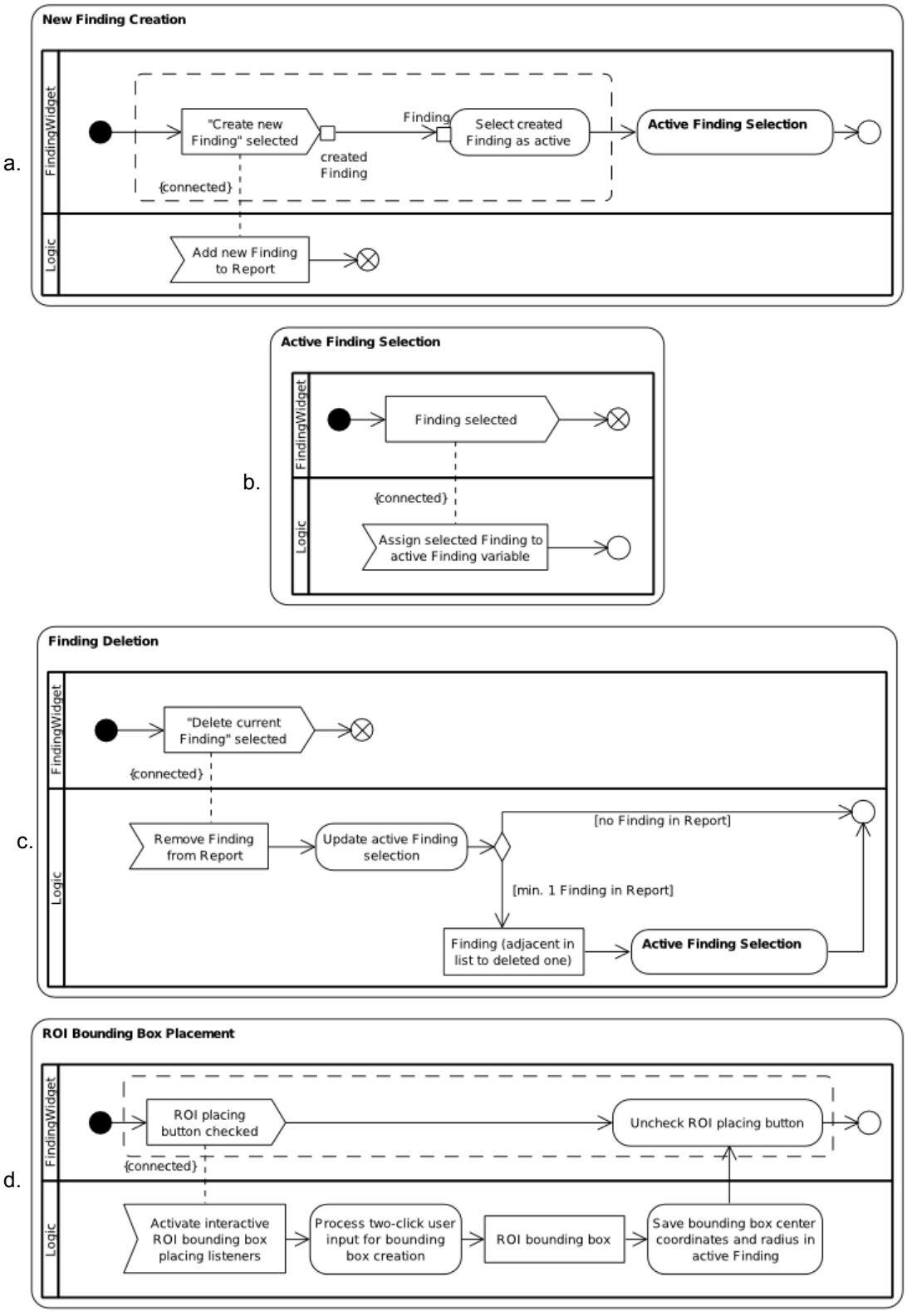


Fig. 3.13: UML activity diagrams for the creation (a), selection (b) and deletion (c) of `Finding` objects and placing of ROI bounding boxes (d) in the third workflow step.

3.3.5 Increment 5 – Segmentation and SUV Calculation

With the introduction of the Finding data structure in the forth increment, the foundation for the actual process of segmentation was laid in the previous build increment. By integrating Slicer's Editor module into the module GUI, access to all advanced semi-automatic and manual segmentation tools that are available in the main application was granted. Since the objective of `Finding` objects is to represent a certain tumor or reference structure within image data from different time points, the module has to provide an interface for performing segmentations on the PET image volumes of each `Study` available in the active `Report`. In the fifth build increment the required data structure, GUI widget and Logic for realization of such interface was implemented.

3.3.5.1 MRML Implementation

Although the data structure for handling of segmentations is provided by Slicer in the form of label map volumes, the implementation of a new class that is capable of storing SUV statistics calculated for such label map volumes was required. For that reason, the new class `vtkMRMLLongitudinalPETCTSegmentationNode` (further referred to as `Segmentation`) was added to the module's MRML classes. Besides the possibility to reference the unique identifier of a label map volume from a specific `Study` object, this class provides 10 further variables. Maximum, mean and minimum SUVs and the standard deviation of those as well as the volume (in cc and mm^3) and total voxel count of the segmentation can be stored in a `Segmentation` object. Furthermore, the center coordinates and the radius of the ROI that has been used to limit the segmentation volume are stored in two three-data arrays similar to the ones used in the `Finding` class. The objective of the last new variable is to store the unique identifier of a 3D triangulated surface model of the segmented volume that is generated upon segmentation.

For the support of the new `Segmentation` class the implementation of `Finding` required enhancement. Hence, a map[32] variable was added, providing the internal data structure for connection of a `Segmentation` object to a specific `Study` object. In doing so, access to the `Segmentation` object, which has been generated for a specific `Study` in the context of a `Finding`, is granted to the module Logic.

[32] Data type composed of a collection of key and value pairs with unique keys

3.3.5.2 GUI Widget Implementation

In contrast to the previous build increment, the interface for segmentation did not require the implementation of a new widget. By integrating the widget representation of Slicer's Editor module into the module's GUI, the requirements for a fully functional user interface for segmentation were automatically met. However, in the workflow of this module segmentations are performed on temporary label maps before they are copied to the target volumes. Hence, an additional button for confirming the segmentation results and triggering the copy process was added to the module GUI underneath the Editor module widget.

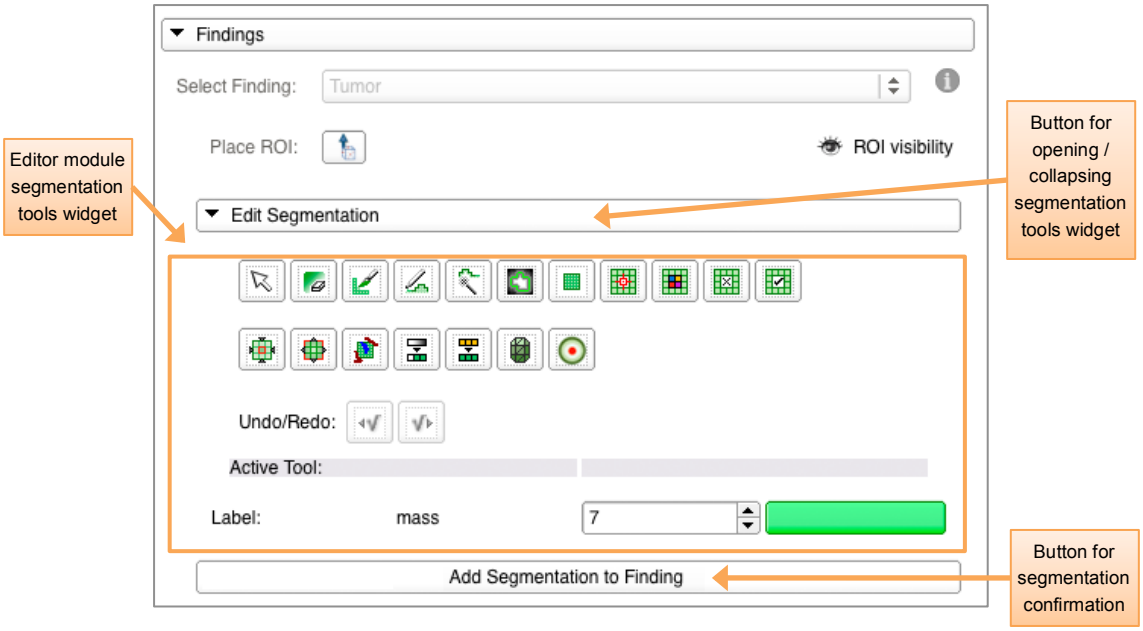


Fig. 3.14: Widget representation of Slicer's Editor module integrated in the GUI widget for `Finding` creation and selection.

3.3.5.3 Logic Implementation

One of the great advantages of reusing the available functionality of the Editor module is the availability of program logic and tools for the process of segmentation. To leverage these possibilities in the module, the Logic is required to prepare the correct volumes for segmentation and afterwards process the results. The segmentation process starts once the user opens the built-in Editor module with its segmentation tools widget (Fig. 3.15a). When receiving the `signal` from the GUI that the segmentation tools widget has been opened, the Logic uses the parameters of the previously defined ROI bounding box to crop the PET image volume according to the ROI extents and zoom in on it in the 2D Slice Viewers. It has to be taken into account that the cropped sub-volume is just a copy of the original one. Hence, a temporary label map volume with the same extents as the cropped volume is prepared for the following segmentation (Fig.

3.15c). As mentioned above, the Editor module provides the logic for the segmentation itself. Once the user has finished the segmentation on the temporary cropped volume, he can confirm and add it to the active `Finding`. When confirmed, the Editor's segmentation tools widget is closed and the GUI emits a `signal` (Fig. 3.15b). This causes the Logic to paste the segmentation results from the temporary label map volume to the target label map volume of the active `Study`. Afterwards, the SUV statistics are calculated and stored in a newly generated `Segmentation` object. This object is then mapped in the active `Finding`. Finally, a 3D triangulated surface model of the segmented volume is generated and displayed in the 3D Volume Rendering Viewer of the main application (see Fig. 3.15c).

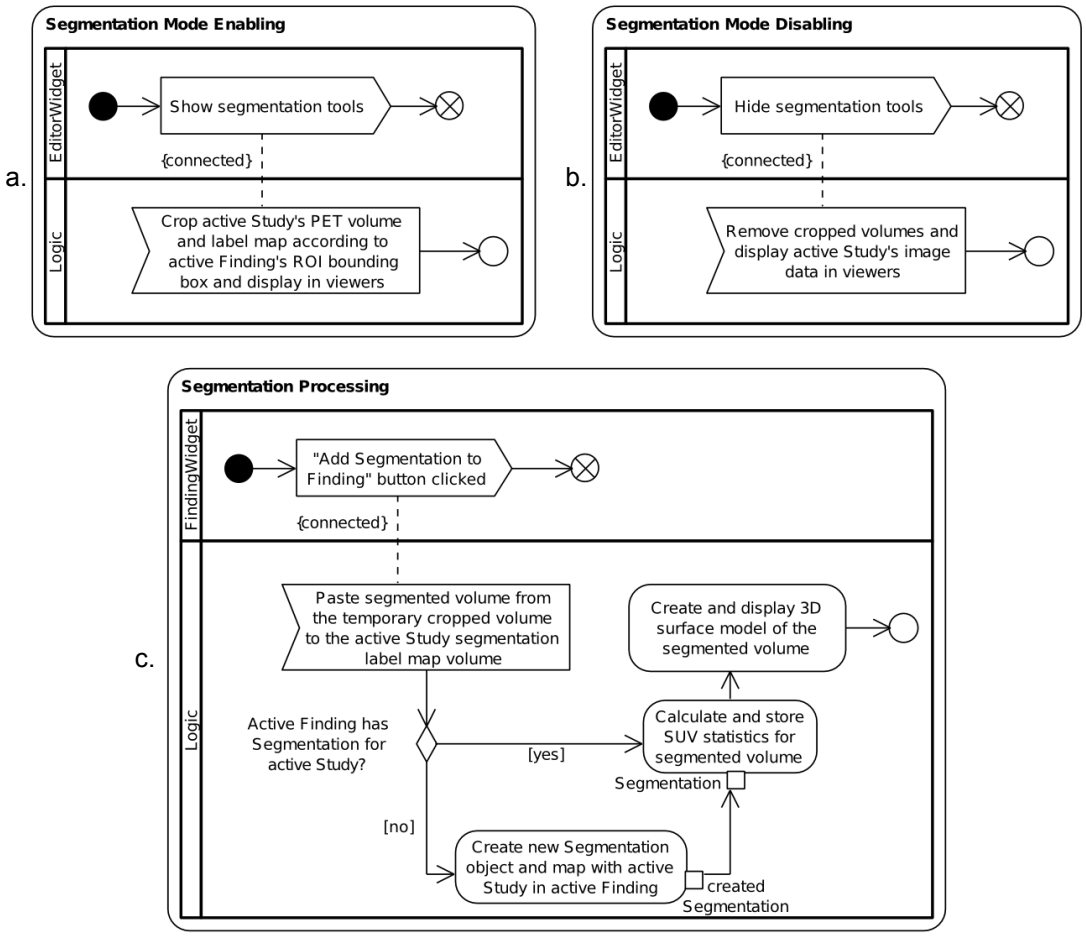


Fig. 3.15: UML activity diagrams for enabling (a) and disabling (b) of the segmentation mode and the actual processing of segmentations initiated from the `FindingWidget` (c).

3.3.6 Increment 6 – Report Overview

Awareness of the current workflow state is a very important aspect when considering the usability of the PET/CT analysis module. In the sixth increment, a tabular widget was implemented providing overview over all available `Study`, `Finding` and `Segmentation` objects in the active `Report`. Moreover, quick selection of active `Study` and `Finding` objects is also possible with the help of this widget.

3.3.6.1 GUI Widget Implementation

The most distinctive feature of the GUI widget for Report overview, which is shown in Fig. 3.16, is its integrated fully interactive table. At the beginning, before `Study` objects have been selected for segmentation or `Finding` objects have been created, the table is empty. However, each time a `Study` object is selected for segmentation, a column is added to the table. Following the same principle, a row is added to the table each time a `Finding` is created. To support quick selection of active `Study` or `Finding` objects, the GUI widget was implemented to emit `signals` when the user clicks on a column or row header.

In order to indicate whether a `Segmentation` object has been generated for a `Study-Finding` constellation, the GUI widget also offers the possibility to fill cells with specific colors. A colored cell indicates that a `Segmentation` object has been generated for a specific `Finding` (represented by the cell's row) on the image data of a specific `Study`, (represented by the cell's column). In addition to that, every colored cell also provides a control that can be used for hiding or showing the 3D surface model of the segmented volume in the 3D Volume Rendering Viewer of the main application. Moreover, cells representing `Segmentation` objects can also display its SUV statistics in a tooltip when the mouse cursor is hovered over the cell. Statistical values are also displayed in the cells themselves but only one type of value can be presented at a time. However, the user has the possibility to define the type of value to be displayed from the selection control in the right upper corner of the widget. At the same time on the opposite side, in the left upper corner, a label is used for providing more specific information about the selected active `Study`. Similar to the GUI widgets implemented in previous increments, an information label is available in the top right corner, providing information about the widget's functionality on mouse hovering.

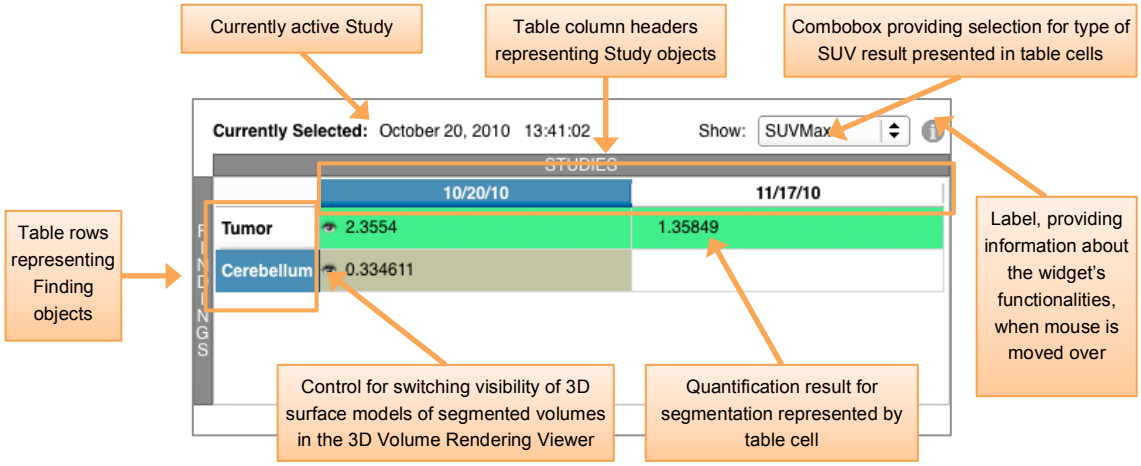


Fig. 3.16: GUI Widget for `Report` overview. In this example the widget presents the overview of a workflow with two analyzed `Study` and two created `Finding` objects.

3.3.6.2 Logic Implementation

Integration of the GUI widget for `Report` overview into the module required the enhancement of the Logic class with three `slot` methods. Two of these are connected to the respective `signals` for change of active `Study` and `Finding`. The third slot is responsible of handling changes in visualization of the 3D surface models of the segmented volumes, which can be displayed or hidden individually to increase the overview in the 3D Volume Rendering Viewer.

3.3.7 Increment 7 – Qualitative and Quantitative Analysis

In the penultimate build increment, the actual analysis functionality was added to the module. Two different types of analysis are realized in the Longitudinal PET/CT analysis module. The qualitative analysis provides the possibility to compare PET/CT image and segmentation volumes side by side in a 2D Slice Viewer grid. For the quantitative analysis, charts of the different statistical values are plotted and visualized using Slicer's built-in chart viewer. For the realization of the seventh build increment, the implementation of a new GUI widget was required as well as the enhancement of the module Logic to handle the changes in visualization.

3.3.7.1 MRML Implementation

Support of individual comparison of analysis results required the enhancement of the `Study` class with an additional `boolean` variable and its access and modification methods. Adding of this variable allows individual labeling of Study objects in order to present or hide their image and segmentation data in the 2D and 3D visualizations of the analysis results.

3.3.7.2 GUI Widget Implementation

The visual appearance of the GUI widget for results analysis is similar to that of the GUI widget for `Study` selection presented in section 3.3.3.2 (see Fig. 3.17). However, the table for `Study` object selection is fulfilling a different purpose here. By checking or un-checking the checkboxes in the different rows of the first column of the table, the `Study` objects represented by those rows are either selected or removed from comparison. In so doing, the widget is offering full support for individual comparison of the analyzed PET/CT studies. Switching between the qualitative and quantitative visualization of the analysis results can be done with the help of the two buttons at the top of the widget. The button at the bottom of the widget is responsible for initiating the export of the quantification results to disk. Information about this widget's functionality can be obtained by hovering the mouse cursor over the label in the top right corner of the widget.

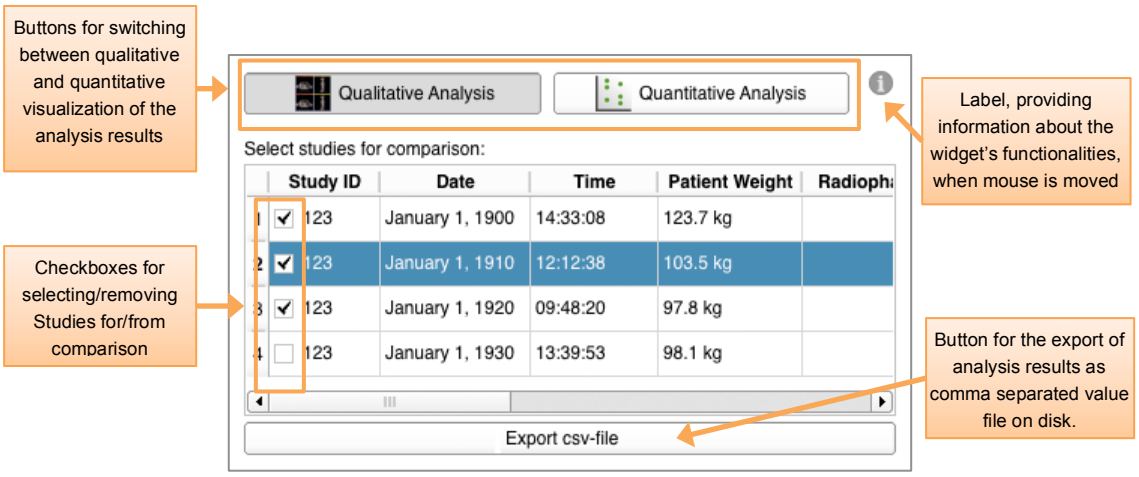


Fig. 3.17: GUI widget for quantitative and qualitative analysis and comparison.

3.3.7.3 Logic Implementation

Enhancement of the module Logic in the seventh build increment included the support of selection/removal of individual analysis results from comparison, switching between qualitative and quantitative analysis visualization and the export of quantification results to disk. When a `Study` object is selected for or removed from comparison a `signal` is emitted by the GUI widget, containing the object itself and the selection state. In the Logic, this `Study` object is modified according to the check state and the method responsible for visualization update is called (see Fig. 3.18c). The same method is also called when the analysis type is changed from qualitative to quantitative or vice versa.

Depending on the selected analysis type, either a 2D Slice Viewer comparison grid for the qualitative (see Fig. 3.18a) or a chart viewer for the quantitative analysis (see Fig. 3.18b) is visualized in the main application. While the 2D Slice Viewer grid displays the PET/CT image data and the corresponding segmentation volumes of all `Study` objects selected for comparison, the chart view is used to present plots of the analysis results. For the export of quantification results to disk, the Logic was enhanced to receive and process the according `signal` from the GUI widget. After a target file is specified with the help of a file dialog, a comma-separated-values file containing the result values is generated and saved to disk (see Fig. 3.18d).

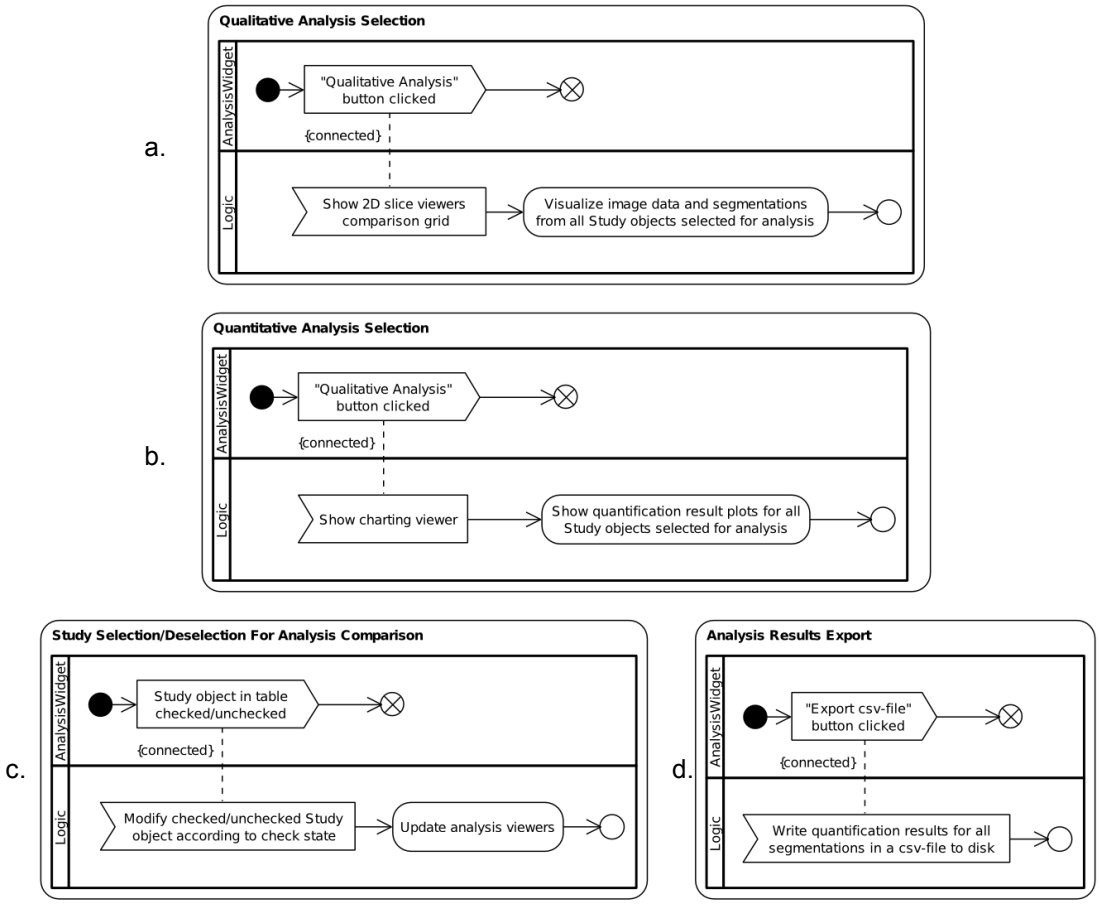


Fig. 3.18: UML activity diagrams for selection of qualitative (a) and quantitative (b), individual comparison (c) analysis and export of results (d) in the fifth workflow step.

3.3.8 Increment 8 – MRML Serialization and Module Settings

Support of serialization and de-serialization of the data generated during a PET/CT analysis workflow as well as its interchangeability between different systems is an important objective of the Longitudinal PET/CT module. Furthermore, automated access to Slicer's various advanced image data visualization settings can be of great help for the analysis workflow and increases the overall usability of the application. The implementation required to meet both above-mentioned goals was realized in the eighth build increment. Besides the enhancement of all MRML classes to support serialization and de-serialization, an additional GUI widget was implemented. This widget contains controls for module specific settings regarding 2D and 3D visualization, availability of volume rendering, 3D surface model generation and registration functionality as well as the interface for integrating external algorithms into the module.

3.3.8.1 MRML Implementation

The implementation of all module specific MRML classes (`Report`, `Study`, `Finding` and `Segmentation`) was completed in this last build increment. To do so, the methods for serialization, de-serialization, duplication and console output of the member variables had to be overwritten. These methods are all inherited from the base class `vtkMRMLNode`. By implementing them, it is ensured that they are compatible with the serialization process of the `MRMLScene`. The same applies for de-serialization of the data when loaded from disk. The methods for duplication of the module's different MRML classes and for console output of their member variables are not necessarily required in the context of this work but were also implemented in order to provide full compatibility with the MRML-Library.

3.3.8.2 GUI Widget Implementation

Usage of Slicer's vast 2D and 3D medical imaging visualization possibilities can beyond any doubt support the user in his decisions during the PET/CT analysis workflow. However, the manual modification of visualization settings can turn relatively quickly into a cumbersome endeavor especially when dealing with numerous image volumes. In order to remedy this circumstances, a GUI widget that can be used as an interface for automated switching between different PET/CT 2D and 3D visualization settings was implemented (see Fig. 3.19). Moreover, details of segmentation presentation, PET/CT image data registration and the method used for PET quantification can also be adjusted in this widget. In contrast to the previous presented GUI widgets, this one's controls follow a vertical layout as they are aligned one below the other. Such layout was intended so that this widget can be used as an always-accessible sidebar panel, positioned besides the GUI widgets for the workflow

steps. The widget is divided into 5 different panels. The top panel provides the controls for PET image volumes 2D and 3D visualization presets. Selection between three different PET specialized color tables is possible. Moreover, controls for enabling or disabling the 3D volume rendering visualization of PET image data as well as triggering a spinning movement of it in the 3D Volume Rendering Viewer and modification of its opacity are provided. In the second panel, the selection from four different Window/Level presets for CT image volumes is supplied. Outlining of segmentations in the 2D Slice Viewers as well as the generation of 3D surface models for segmented volumes can be enabled or disabled in the forth panel. The fifth panel is concerned with the settings for registration of PET/CT image data in the analysis workflow. Finally, the last panel is the quantification settings panel. By clicking on the button in this last panel, a text input dialog is opened, offering specification of the CLI module that is responsible for the quantification of the loaded DICOM PET data. By using this interface, external developers can exchange the default PET quantification method of the module with their own algorithms.

3.3.8.3 Logic Implementation

Serialization and de-serialization of the module's MRML classes is handled automatically by the main application, hence it did not require adaption of the module Logic. However, providing functionality for the module settings GUI widget required enhancement of the module Logic. New methods were implemented in order to be able to receive the various `signals` emitted by the GUI widget. Besides updating the variables, which are affected by setting modifications, the Logic also has the objective to save the current state of the setting controls to disk. This way the module can be started every time with the exact same settings it was closed after a previous execution. When stored on disk, unique identifier for the different settings and their respective values are written into a text-based file, which is provided by the main application for such purposes.

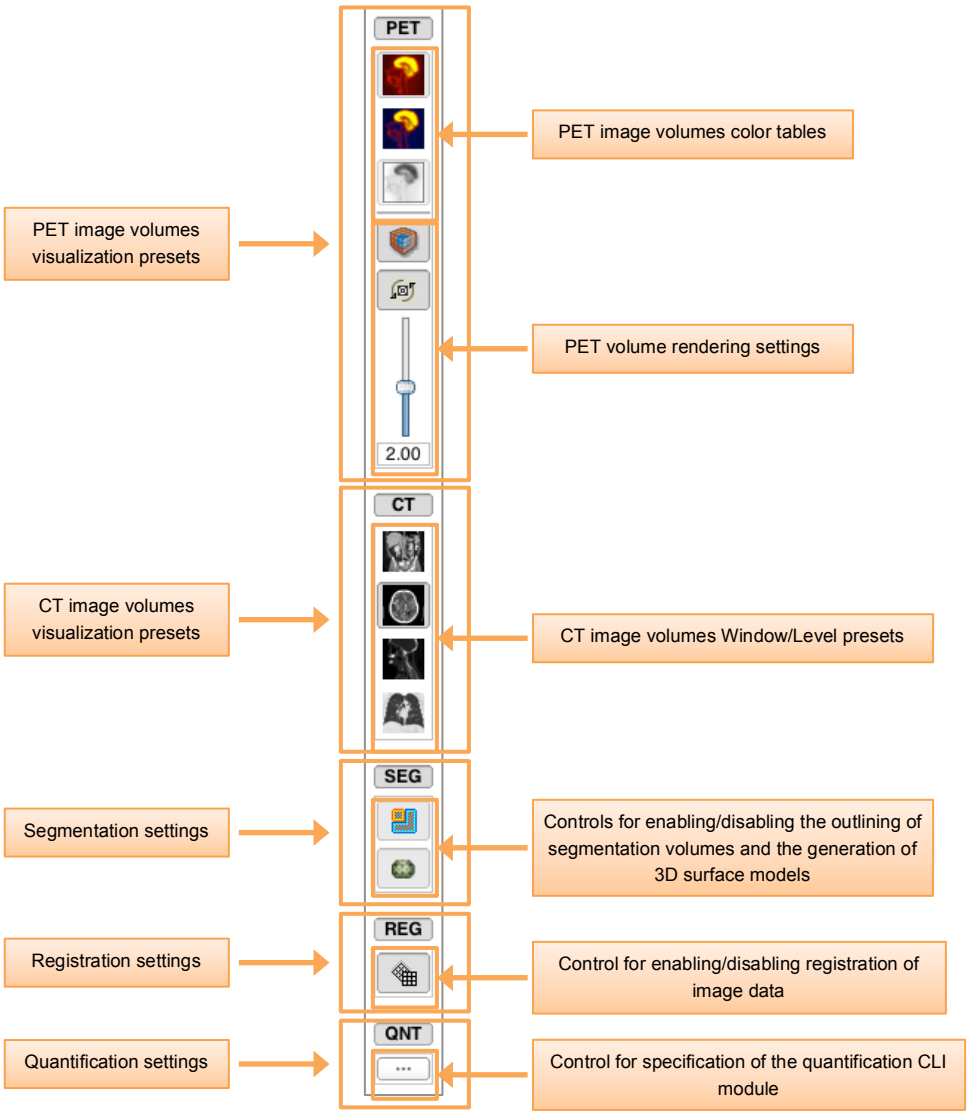


Fig. 3.19: Module settings GUI widget.

3.4 Evaluation

Following the finalization of a first stable release, efficiency of the workflow, reproducibility of analysis results and usability of the implemented module had been evaluated. The methods used for the evaluation of the listed parameters are presented in this section.

3.4.1 Keystroke Counts Comparison

Counting the keystrokes (in the following, the term keystroke is also used to describe mouse-clicks) that are required to perform a certain task is a well-suited method to measure the efficiency of a software workflow. In the context of the keystroke counts evaluation study presented here, the module realized in this work was compared to the PET/CT Fusion module from 3D Slicer 3 and the PET Standard Uptake Value Computation module from the current version. While the former is the PET/CT analysis module of the previous version of Slicer, the latter is a CLI module capable of single study PET quantification in the current version of Slicer. Comparison with the PET/CT Fusion module, which also has been examined in this project's requirements analysis, helped to reveal whether the planned improvements for PET/CT analysis in Slicer could be achieved. Comparison with the PET Standard Uptake Value Computation module showed if PET analysis in a guided workflow is superior in efficiency to manual analysis using a CLI module.

The protocol of the keystroke counts comparison in this work stipulated that three predefined tasks for PET analysis (see Table 3.4) had to be executed with each of the PET analysis modules. Based on the nature of these tasks, the amount of analyzed PET/CT studies could be included as a factor (n) into the keystroke counts formula and did not have to be determined beforehand. In so doing, the suitability of the different modules for longitudinal analysis could also be taken into account.

Table 3.4: Keystroke counts evaluation tasks.

Task 1:	Loading of *n* PET/CT DICOM Studies from Slicer's DICOM database.
Task 2:	PET/CT fusion visualization in 2D Slice Viewers with PET and CT image volumes from *n* loaded studies.
Task 3:	Quantification of a tumor and reference VOI using a predefined label map volume for each of the *n* PET volumes.

3.4.2 Quantification Results Reproducibility

Quantitative assessment of tumors is less evaluator-dependent than qualitative assessment, as already mentioned in chapter 2.1.5. However, as long as it is not performed with fully automatic tools, quantitative assessment is still influenced by subjective factors. In addition to this circumstance, the experimental protocol of SUV computation is quite simple and far away from being a precise method for PET quantification. Nevertheless, to this day it is the only method that is applied for PET quantification in clinical practice. For that reason, criteria for treatment response assessment focusing on tumor SUV changes, e.g. from the European Organization for Research and Treatment of Cancer (EORTC), is based on empiric cutoff points (15%, 25%, 50%) that categorize in which order of magnitude the metabolic activity has decreased, increased or if it remained stable (< 15% change in tumor SUV) [61].

The goal of the assessment of quantification results reproducibility in this work was to evaluate the dimension of variability in PET analysis results from different evaluators. Based on the results from this evaluation it could be estimated whether the longitudinal analysis results produced by this work are precise enough to be applied to the above-mentioned criteria for treatment response assessment. The setup of this evaluation study contained of the following.

- Evaluators: 8 evaluators, not specialized in PET assessment
- Hardware: Apple Macbook Pro (Dual Core i7, 2.7GHz, OSX 10.7)
- Dataset: 3 DICOM FLT-PET/CT head studies with brain tumor

All evaluators who participated in this evaluation had to perform a 3-timepoint longitudinal PET analysis by using the workflow provided by the Longitudinal PET/CT module. For each study, the brain tumor had to be segmented in the PET volume by using the built-in segmentation tools of Slicer. The primarily used tool was the threshold segmentation, which in general obtains good results on PET images due to the high intensities of metabolic active tissue. Furthermore, a semi-automatic tool for removal of segmentation fragments not belonging to the segmented tumor (caused by the previous threshold application) has been used as well as a manual drawing tool for segmentation. Eventually, the computed minimum, mean and maximum SUVs of the tumors segmented at all three time points have been documented for each evaluator.

3.4.3 Usability Evaluation

The usability of the implemented module was validated with a combination of heuristic evaluation, questionnaires and personal interviews. Two different groups of evaluators were involved in this evaluation. While one group was composed of clinical experts in the field of PET research, Slicer development experts formed the other group. Both groups were asked to apply usability heuristics to the PET/CT analysis workflow of the module. Feedback from the clinical experts group was collected in personal interviews after a detailed presentation of the workflow. For recording of feedback from the software developer group, questionnaires were handed out. These questionnaires contained free form text fields for notes concerning usability violations and additional questions designed to increase awareness of the most important heuristics (see appendix D). The use of two different experimental protocols was due to the limited time clinical experts could spare from their everyday work for this evaluation. However, as already mentioned in chapter 2.5.2, heuristic evaluation can be performed with very few evaluators. Studies have shown that approximately 70% of usability problems in a system can be found by heuristic evaluation with 4 and approximately 80% with 7 evaluators [44]. These results confirm that even with two different experimental protocols for the two different evaluator groups, the heuristic usability evaluation conducted in this work had the potential to reveal at least 70% of usability problems.

To provide an overview for the usage of the module, a screen-cast of a complete PET/CT analysis workflow has been recorded and made online accessible for all evaluators [62]. The complete setup of the evaluation is presented in Table 3.5.

Table 3.5: Heuristic usability evaluation study setup

	Clinical experts	**Slicer development experts**
Group size	3	4
Preparation		Online video with workflow demonstration.
Execution of the module	Presentation of the workflow by the facilitator following the same execution procedure as in the online screencast.	Hands-on usage of the module by each evaluator.
Recording of usability violations	Personal interview with the facilitator during presentation of the workflow.	Questionnaire with free form text fields and additional questions for every workflow step.
Final severity rating	Reported usability violations from all clinical expert evaluators are summarized in a form and sent back for individual severity rating.	Reported usability violations from all software development experts are summarized in a form and sent back for individual severity rating.

4 Results

In the following, an overview of the appearance and functionality of the GUI prototypes implemented based on the results of both requirements analysis iterations is given. Furthermore, this chapter is concerned with the results of the conducted validation studies.

4.1 Mockups for the workflow based on initial requirements

4.1.1 Workflow Step 1: DICOM Import and PET/CT Study Selection

The intention of the first workflow step as to manage the import and selection of PET/CT data required for further segmentation and quantification. The module accepts only PET and CT data imported from Slicer's DICOM database since quantification of PET data can only be realized with additional information from its DICOM header. Once a patient from the database is selected, the table is populated with entries of his/her different PET/CT studies (see Fig. 4.1). By checking or un-checking the box in the first column of a study entry in the table, the respective study is added to the workflow or removed from it. Once a study entry is checked, its PET and CT image data is loaded into the 2D Slice Viewers and the 3D Volume Rendering Viewer on the right-hand side. Fusion imaging and 3D volume rendering are used for the visualization. Based on the selection, the slider below the table is updated and shows the different acquisition time points of selected studies. Further, this slider allows a quick switching between the actually selected studies. Whenever a study in the workflow is selected by using the slider, the corresponding image data is shown in the viewers.

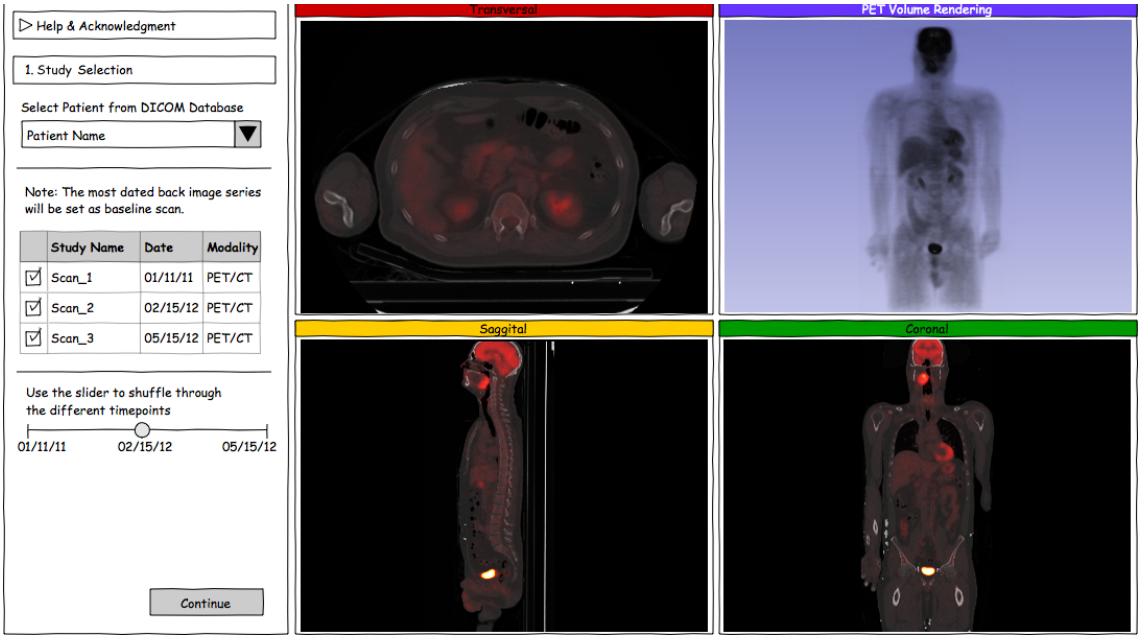


Fig. 4.1: Mockup for the DICOM import and study selection step of the workflow.

4.1.2 Workflow Step 2: Defining Regions of Interest

Narrowing down interesting structures for segmentation requires the use of regions of interest (ROIs). Organization of ROIs from the same or different structures across different studies can be implemented by using a three-leveled tree hierarchy. On the first level, a root node is representing the current workflow. The third level consists of the leaf nodes for the different ROIs. In between, the middle level of the tree is represented by a data structure, grouping ROIs, which are localizing the same structure at different time points. An example of the overall structure is shown in the tree view in Fig. 4.2.

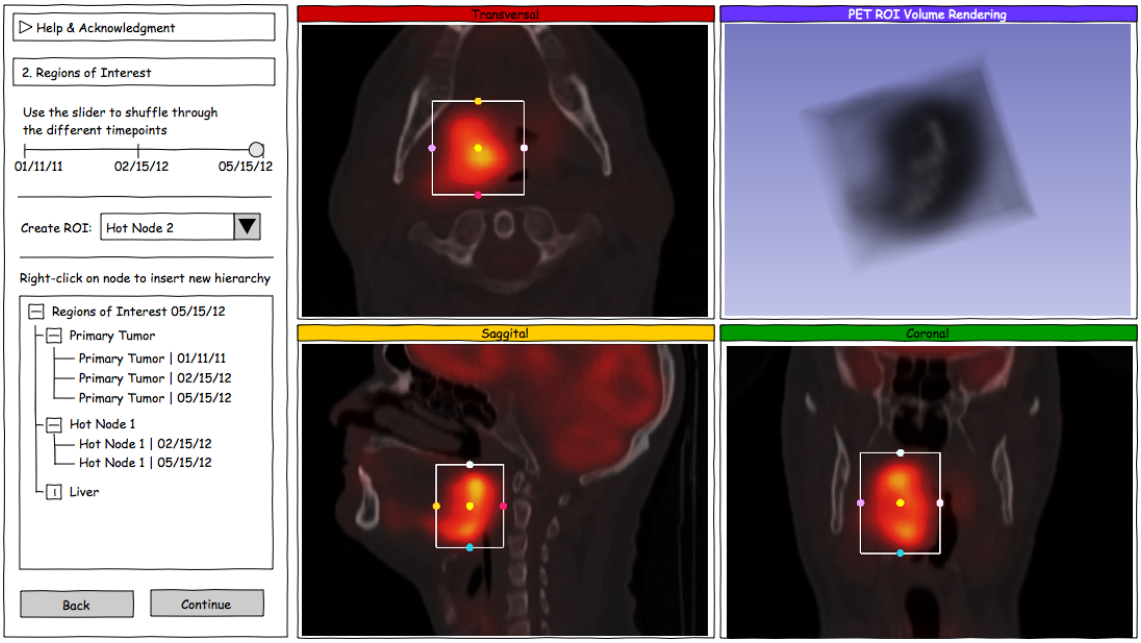


Fig. 4.2: Mockup for the ROI definition step.

4.1.3 Workflow Step 3: Segmentation

To achieve the second goal of this work, namely to create an extensible platform for the support of further developments in PET specialized registration, quantification and especially segmentation algorithms, the suggested segmentation possibilities in the mockup shown were kept rather simple. Therefore, only a double slider control for basic gray scale value based threshold segmentations was integrated (see Fig. 4.3). The need of various and diverse segmentation methods for the tool to be developed was evident. However, by presenting only a single segmentation method in this throwaway prototype the objective was pursued to improve the elicitation regarding essential and desired segmentation tools. In addition to the visualization in the 2D Slice Viewers, a 3D surface model of the current segmented volume of interest is presented in the 3D Volume Rendering.

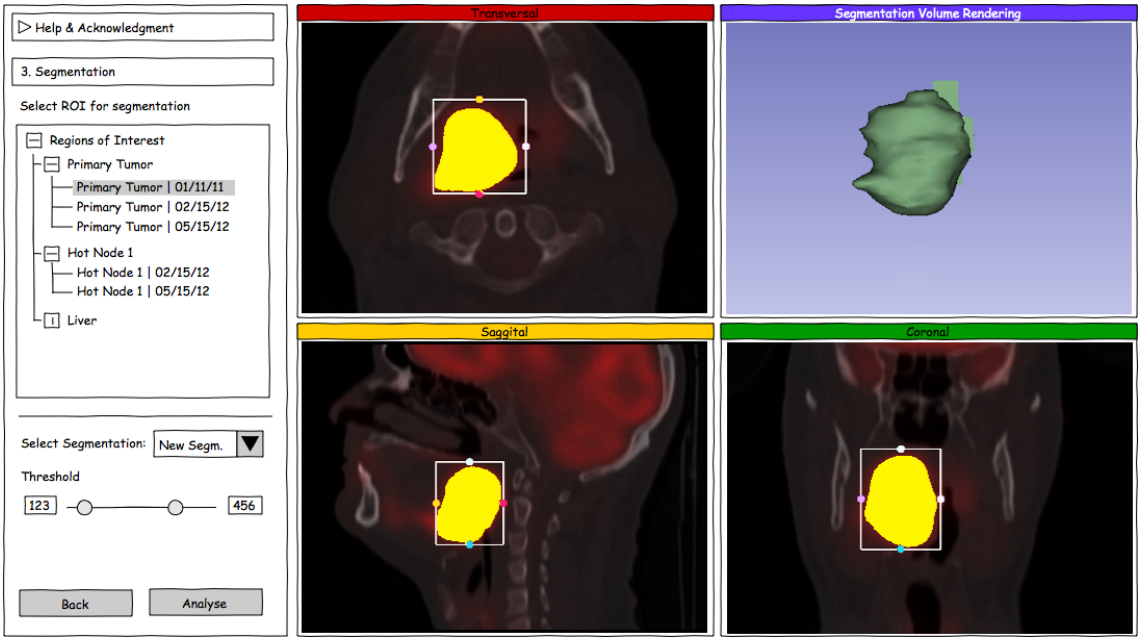


Fig. 4.3: Mockup for the segmentation step.

4.1.4 Workflow Step 4: Analysis

The last step in the workflow is the overall analysis of created segmentations and automatically computed quantification results. Two different types of views are required in this step since the visual inspection of segmentations is of a qualitative nature, whereas the analysis of uptake values is of a quantitative one. Visualization for the qualitative analysis is obtained by aligning the 2D Slice Viewers of each study next to each other thus allowing comparison at a glance. These viewers are aligned horizontally as shown in Fig. 4.4. To avoid an overflow of the screen when numerous studies have been added to the workflow, the individual selection of studies for comparison purposes is possible. This can be achieved by modifying the selection of the entries in the upper table, which is similar to the one from the first workflow step.

Representing the computed quantification results in chart viewers is the main characteristic of the quantitative analysis view (see Fig. 4.5). Each of the vertically aligned viewers stands for a different study. Consistent to the functionality of the controls in the qualitative view, the individual selection of studies to be compared removes or adds the respective chart from or to the comparison. Below the chart viewers a summary of the workflow is presented, containing all analysis results obtained during the workflow.

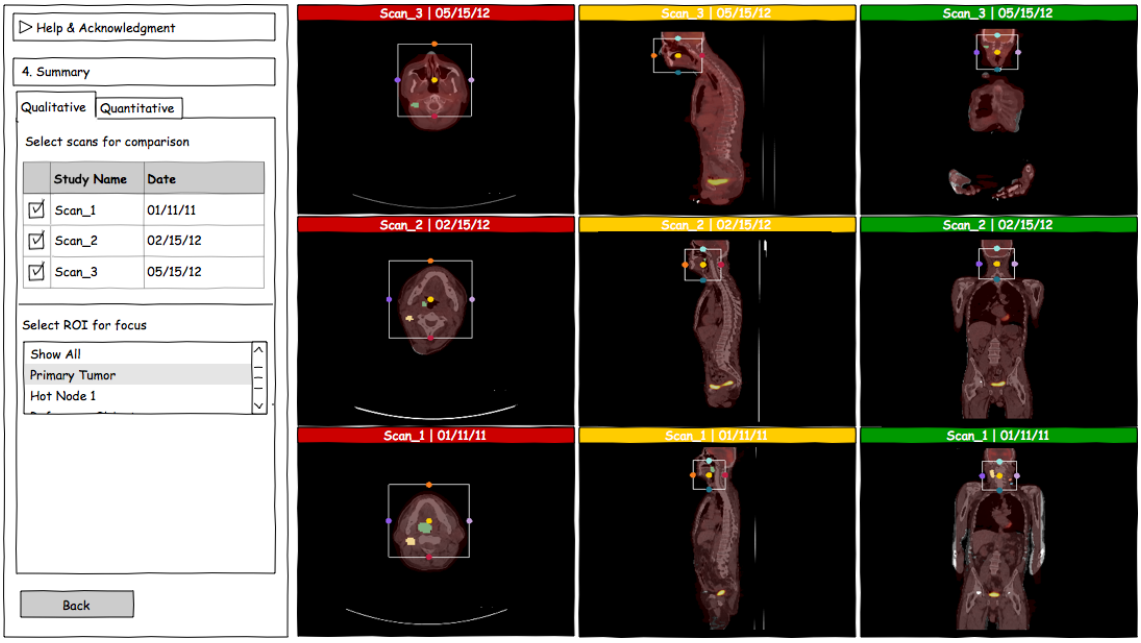


Fig. 4.4: Mockup for qualitative analysis view.

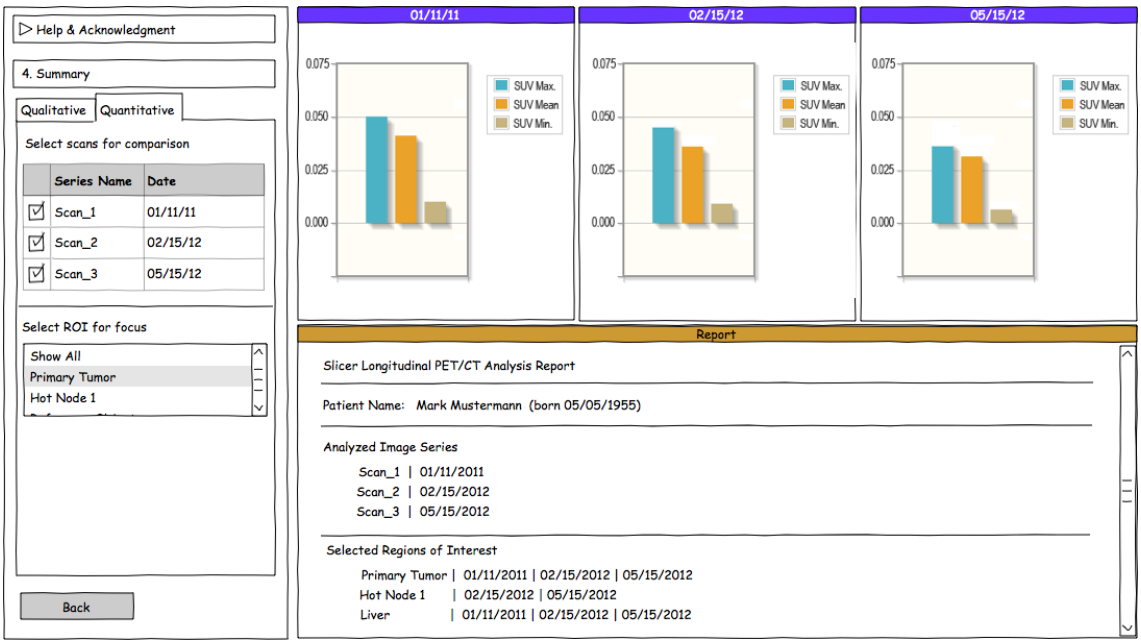


Fig. 4.5: Mockup for quantitative analysis view.

4.2 GUI Prototype Demo Application

4.2.1 Prototype Refactoring

In order to improve the understanding for the overall operation of the planned software, a second prototype with a fully interactive GUI and dummy data for the demonstration of an example workflow was implemented subsequent to the second iteration of the requirements analysis process. Screenshots of this prototype and its appearance in the different stages of the example workflow are presented in Fig. 4.6 and briefly explained in the following.

At the beginning of every workflow a “Report” has to be created. Information about all data imported into or generated in a workflow are stored within this Report. Another noticeable change is the sacrifice of the wizard driven workflow organization. Each workflow step can be made visible with just one click, provided that all data required for the step to switch to is available. Since each step depends on the data prepared in the previous steps, unrestricted switching is only possible as soon as the selection of the last step is enabled.

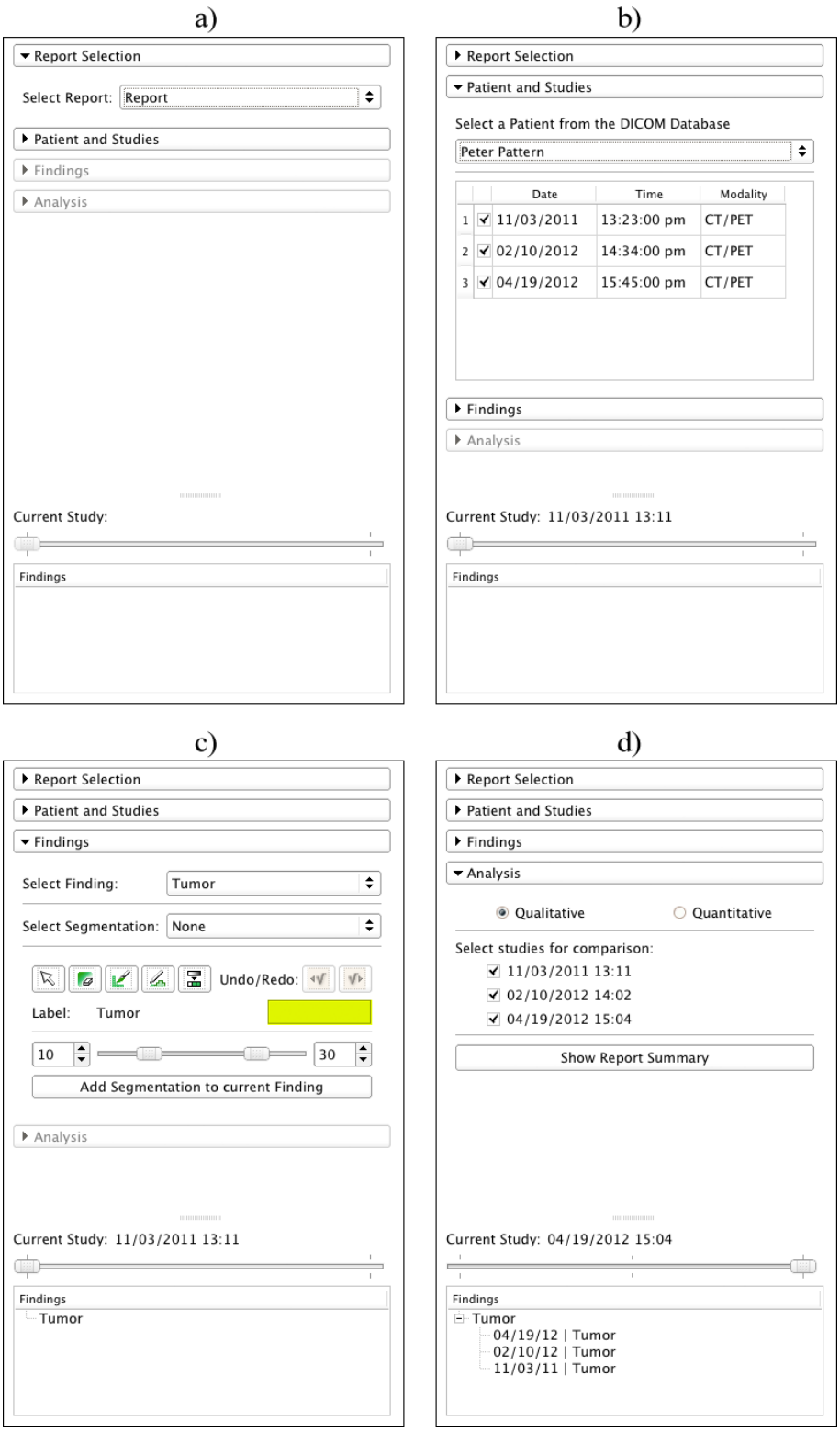


Fig. 4.6: Different workflow steps of the interactive GUI prototype: Report selection (a); Study selection (b), Finding creation/selection and segmentation (c), Analysis (d).

By design, only the GUI controls required for one workflow step are visible at a time. However, the slider for time point switching and the tree view for the “Finding” data structure have been detached from certain workflow steps where they were located in the first GUI mockups prototype. Both are permanently visible and maintain the same position in the user interface. Moreover, they offer a very convenient way for quick selection of specific studies and segmentations. The placement of ROIs has been bundled with the creation of Findings. The idea behind this approach was that every time a Finding is created, an ROI of predefined size is placed at the crossing point of the three 2D planes shown in the Slice Viewers. This ROI can then be adjusted manually. In Fig. 4.6 c), the controls of the third workflow step, where Findings are created or selected and segmentations can be performed using Slicer segmentation tools, are shown. Each Finding can contain no more than a segmentation per time point. GUI controls for the qualitative and quantitative analysis step of the workflow have not changed significantly, whereas the analysis visualization is planned to offer *High-Low-Close* chart types and multiple 3D volume rendering viewers (see Fig. 4.7).

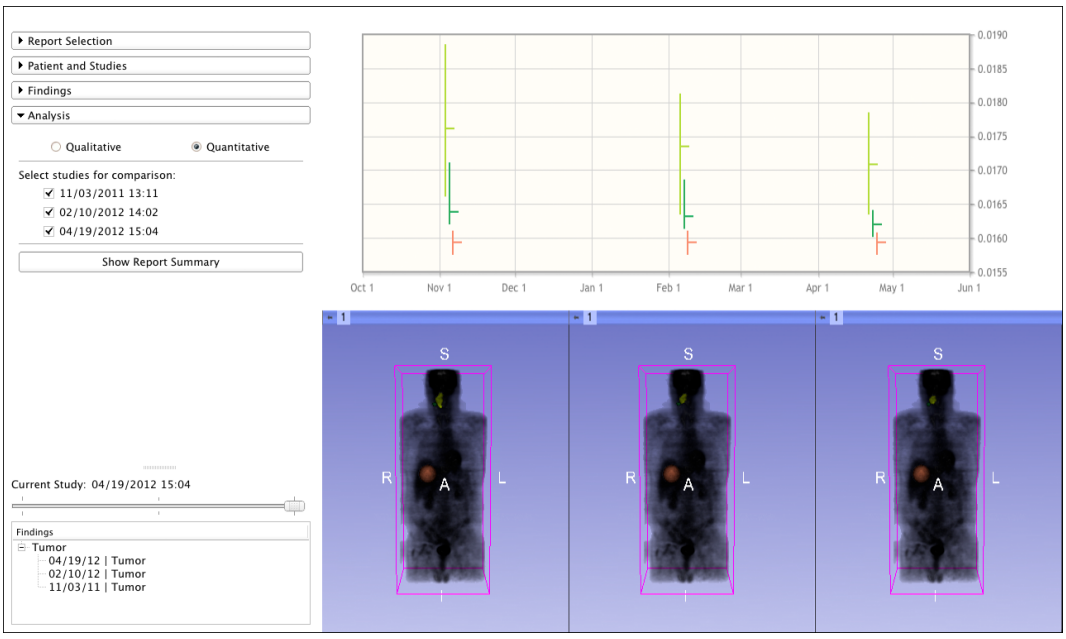


Fig. 4.7: Visualization of the quantitative analysis including HLC-Charts for SUVs plotting and 3D volume rendering views of PET data and segmentations from every study / time point.

4.3 Keystroke Counts Comparison Results

4.3.1 Loading of PET/CT DICOM Studies

Based on the fact that loading of PET/CT studies from Slicer's DICOM database is handled by a plugin in the newly implemented module (Longitudinal PET/CT Analysis), the number of keystrokes required for import is independent from the amount of loaded studies. In contrast to that, the keystroke counts for loading of PET/CT DICOM studies without the plugin (for the PET Standard Uptake Value Computation CLI module) shows a linear increase due to manual selection of the PET and CT image series for each study. The most keystrokes are required for loading PET and CT DICOM image series with the PET/CT Fusion module (see Fig. 4.8). This is mostly based on the fact that Slicer 3 does not provide a DICOM browser for quick and easy access to DICOM data, thus loading of image series has to be done manually by importing the image files located on disk.

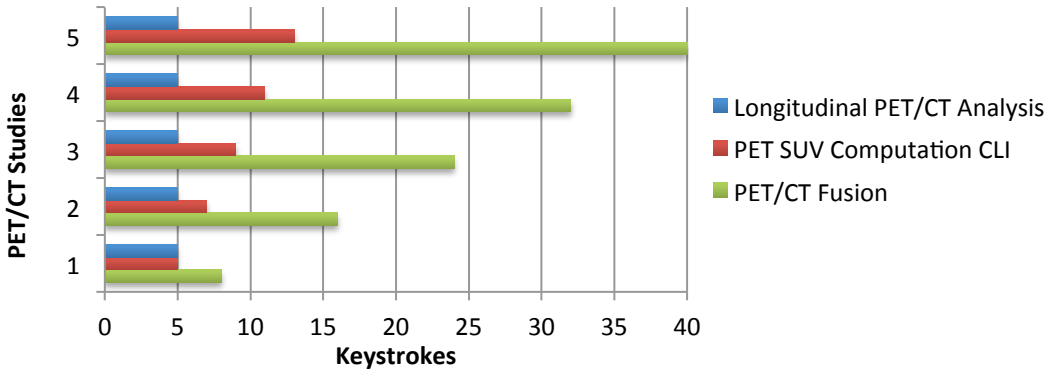


Fig. 4.8: Results of the keystroke counts comparison for the first task: "Loading of n PET/CT DICOM Studies from Slicer's DICOM database".

4.3.2 PET/CT Fusion Visualization in 2D Slice Viewers

Automatic setup of PET/CT fusion visualization upon selection of a study in the Longitudinal PET/CT Analysis module limits the amount of keystrokes for such visualization to the amount of loaded PET/CT studies. However, the PET Standard Uptake Value Computation CLI module does not provide such selection and requires a cumbersome assignment of color lookup-tables for PET volumes and the manual selection of foreground and background images in the 2D Slice Viewers for each study. For that reason, the amount of counted keystrokes is significantly higher (see Fig. 4.9). As the name implies, the PET/CT Fusion module is more suitable for PET/CT fusion visualization, which is also reflected in its keystroke counts results for this task.

However, its results underperform compared to the newly implemented Longitudinal PET/CT Analysis module, especially as the number of studies is increasing.

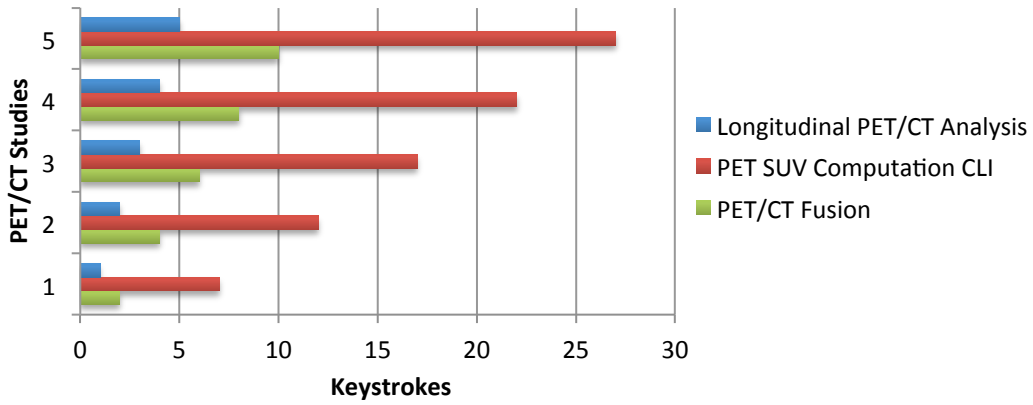


Fig. 4.9: Results of the keystroke counts comparison for the second task: "PET/CT fusion visualization in 2D Slice Viewers with PET and CT image volumes from n loaded studies".

4.3.3 PET Quantification

Quantification of VOIs segmented from PET image data is the main purpose of all compared modules. However, only the Longitudinal PET/CT Analysis module was especially designed to facilitate longitudinal analysis. As seen from the diagram in Fig. 4.10, the keystrokes required for single study quantification are equal or, respectively, differ by one. A noticeable difference only occurs when VOIs from multiple PET images are quantified. In this case, the Longitudinal PET/CT Analysis module has shown to perform better.

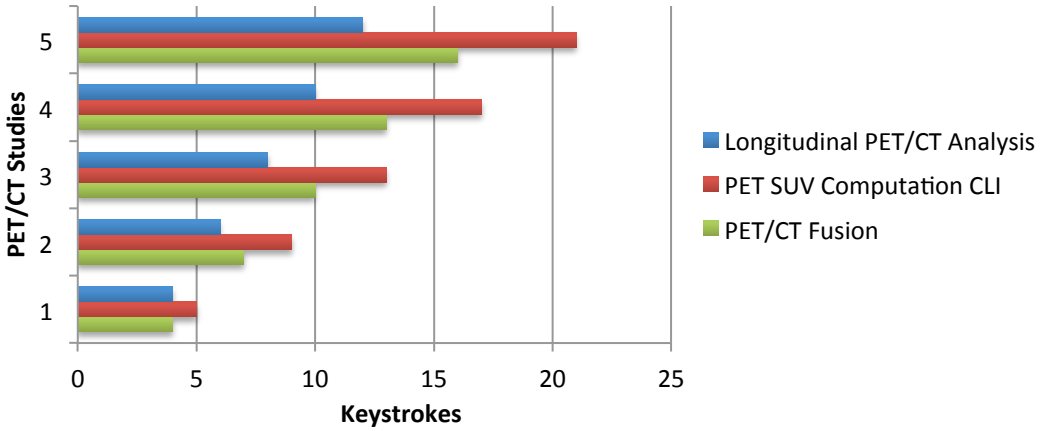


Fig. 4.10: Results of the keystroke counts comparison for the third task: "Quantification of tumor and reference VOIs using a predefined label map volume for each of the n PET volumes".

4.4 Quantification Reproducibility Results

The results of the quantification reproducibility study demonstrate the robustness of the quantification method used in this work and to which extent the preceding evaluator-dependent segmentation process influences its results. All evaluators were able to generate the same maximum SUV for the tumors in all respective datasets. However, this optimal result is mostly due to the simple experimental protocol of maximum SUV estimation, in which the highest quantified segmented voxel represents this maximum value. On the contrary, the mean SUV derives from the totality of voxels in the segmented VOIs, thus is directly influenced by the individual segmentations of the evaluators. For that reason, reproducibility of mean SUV exhibited variability between 3.67% and 4.69% in relation to the maximum tumor SUV of the baseline dataset and between 4.69% and 10.31% in relation to the maximum tumor SUV of each respective study (see Fig. 4.11 left). Similar to the estimation of maximum SUV, the minimum SUV in a VOI is represented by the lowest quantified voxel. However, the results in this study show that the minimum SUV could not be reproduced as optimal as the maximum SUV. This is mostly due to the fact that it is more likely to segment the highest quantified voxel, which is usually somewhere within an area of high SUV values, than the lowest quantified voxel, which will rather be found on the periphery of the VOI. In this study, minimum SUV reproducibility results exhibited variability between 2.54% and 4.26% in relation to the maximum tumor SUV of the baseline dataset and between 4.26% and 8.91% in relation to the maximum tumor SUV of each respective study (see Fig. 4.11 right).

Regarding the results that were normalized with the respective tumor SUV_{max} of each study, it becomes apparent that an increase of variability takes place from one dataset to the other (see Fig. 4.11 right). This trend is due to the characteristics of the image data used in this study. The overall tumor SUV is decreasing between the baseline and the follow-up studies (based on successful treatment). For that reason, the contrast of SUV values in the tumor region is reduced. Hence, differentiation between healthy tissue and tissue with increased metabolic activity becomes more complicated, which leads to more variations in the evaluator-dependent segmentation process. However, this study has been conducted with evaluators who were not specialized in PET assessment, thus it can be assumed that evaluators with expertise in PET analysis will most likely be able to achieve less varying results.

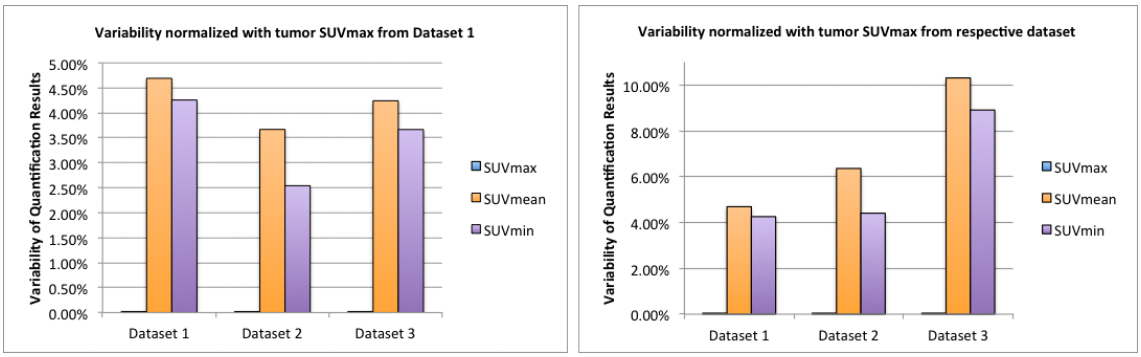


Fig. 4.11: Results from quantification reproducibility evaluation with 8 evaluators and 3 head PET/CT image datasets, normalized with tumor SUV_{max} of Dataset1 (left) and tumor SUV_{max} of each respective dataset (right).

4.5 Heuristic Evaluation Results

The results of the heuristic usability evaluation are of a twofold nature. While the Slicer development expert group reported some minor usability problems, mostly related to the GUI, feedback from clinical experts group was focused more on additional desired capabilities. In the following, results from the usability problems severity rating and the feature request rating are presented.

4.5.1 Slicer development experts usability problems severity rating

The Slicer experts group used questionnaires to report usability violations. Inspection of this feedback revealed in total seven usability problems that the evaluators encountered during execution of the module. These problems were:

1. Information labels in the top right corner of workflow step GUI widgets are too plain when colored gray.
2. Access to the DICOM browser is only possible through the Slicer menu and not offered by the module's GUI.
3. The order of the GUI widgets for the subsequent workflow steps is not necessarily obvious from their arrangement.
4. Loading of just the result presented by the PET/CT DICOM plugin always requires deselection of the additional results from other plugins that are selected by default too.
5. Default enabled spinning movement of 3D volume rendering is remarkably slowing down the performance of the system.
6. It takes a little to get used to the two-click ROI placing mechanism.
7. The Report overview is not able to present all quantification results at once.

The severity of each one of the 7 summarized problems was rated by all 4 evaluators of the group using a severity rating scale for usability heuristics [48]. The result of this ranking demonstrates that from the 7 usability problems only 2 were considered to be real usability problems, while the others were rather ranked in between cosmetic and minor usability problems (see Fig. 4.12).

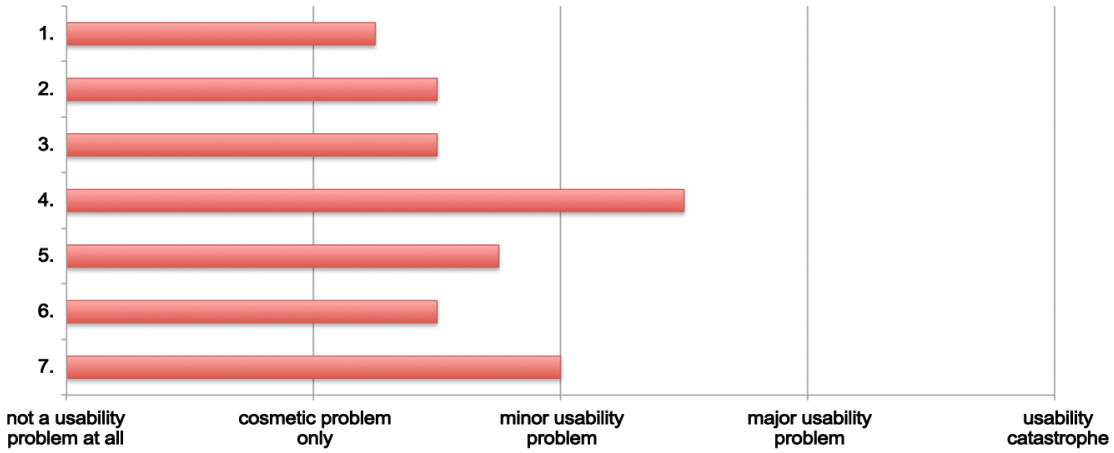


Fig. 4.12: Severity raking of the 7 summarized usability problems, performed by 4 of 4 evaluators of the Slicer developer experts heuristic evaluation group.

4.5.2 Clinical experts feature request rating

Usability feedback from clinical experts was collected during personal interviews. As mentioned above, this feedback was more focused on usable features that are not yet implemented in the module than on usability problems. For that reason, all reported feature requests have been summarized to 6 feature requests:

1. Calculation of SUV_{peak}[33]
2. Threshold segmentation tool based on percentage of SUV_{peak}
3. Visualization of complete PET DICOM header
4. Manual (rigid) image registration across timepoints
5. Visualization of tumor growth/shrinkage comparison between studies
6. Kinetic modeling support for dynamic PET imaging

The clinical expert evaluators (2 of 3) rated the importance of each of these features for an elaborated PET analysis workflow. The majority of the requested features were

[33] More robust quantification method than SUVmax, in which a predefined ROI is placed around the maximum SUV or highest uptake region and the average SUV in this ROI results in the SUVpeak.

rated as important enhancements or even crucial requirements for a correct PET analysis workflow (see Fig. 4.13).

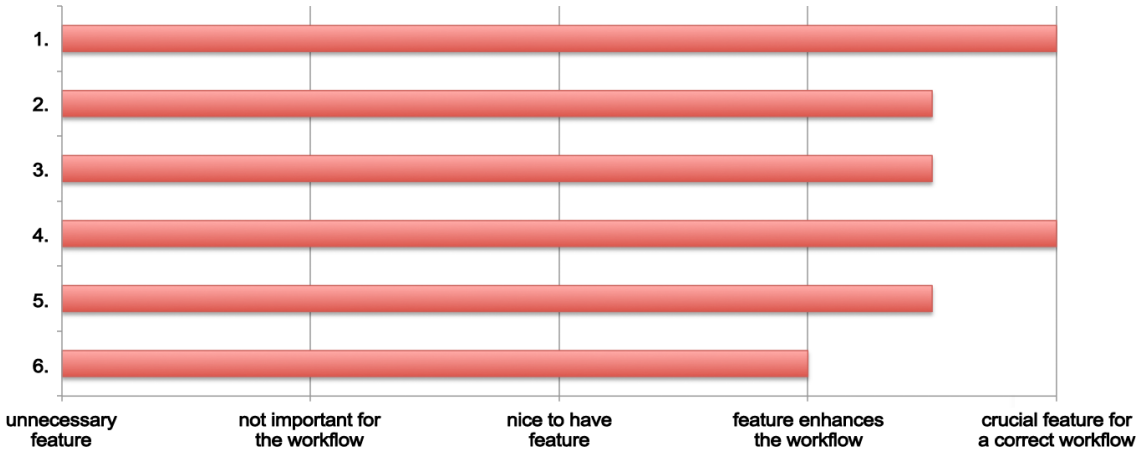


Fig. 4.13: Feature importance raking of the 6 summarized feature requests, performed by 2 of 3 evaluators of the clinical experts evaluation group.

5 Discussion

PET imaging has become a very important tool for oncology due to its potential to visualize metabolic activity and is increasingly used in clinical practice for diagnosis and staging of tumors. However, assessment of PET is predominantly carried out through simple visual inspection. Even though the importance of PET quantification is generally accepted, only simplified measures like SUV are used in routinely diagnosis and staging.

In this work, a software tool has been developed with the goal to fulfill two purposes. First, a workflow is provided for the analysis of time-series PET image data. During the implementation of the tool, significant efforts have been made to ensure that said workflow is intuitive and efficient to use even for operators who are not specialized in the field of PET analysis. Second, interfaces were integrated into the software architecture in order to allow fellow researchers to use this tool as a platform for the prototyping and development of new PET specific quantification and segmentation algorithms. When extending the software with such algorithms, developers benefit from the fact that they can build up on the available workflow. This way, their focus lies on prototyping while data handling and visualization is provided by the underlying platform and supported by the workflow. Moreover, the tool implemented in this work is also well suited for validation of new quantification and segmentation methods by clinical evaluators.

Validation of the software tool realized in this work has shown that an improvement in operational efficiency, compared to alternative tools for PET quantification in Slicer, has been achieved. Unfortunately, a similar efficiency comparison with a commercial system was not feasible in this work. Nevertheless, live demonstrations of PET quantification workflows in commercial systems (from *Siemens* and *Hermes Medical Solutions*) proved that there is no significant difference in efficiency between those tools. However, the commercial solutions offer more sophisticated segmentation and quantification methods than the ones provided in this work.

Taking a closer look at the usability evaluation results, it becomes apparent that the prototype driven implementation of the software had a positive effect on the achieved usability. None of the evaluators reported a severe usability issue, which e.g. would bear the potential to make the software unusable for PET analysis. However, feedback from the clinical experts indicated that a few additional PET segmentation and quantification features need to be implemented in the future, in order to bring the tool on a similar level of functionality as it is offered by commercial solutions.

In the quantification reproducibility validation study, an average variability of 3.84% was achieved when variations were considered in relation to the maximum SUV of the

baseline study. On that account, it can reasonably be concluded that the precision of quantification results offered by this work suffices for a basic assessment of treatment response. In particular because such assessment is usually done by comparison of quantification results from follow-up studies with the baseline-study, and empirical cutoff points for categorization are chosen relatively wide (15-25%). However, intra-study mean and maximum SUV variability reached up to 10.34%, which is rather high. Based on the fact that the reproducibility study was performed by evaluators who were not experts in PET analysis, it can be concluded that most of the variability is due to missing expertise in PET analysis rather than to inaccuracies in the quantification methods. Future validation studies performed by evaluators with expert knowledge in PET analysis will have to show if intra-study mean and maximum SUV variability can be reduced to an adequate level.

Probably the most relevant drawback of the implemented work at this stage is the lack of peak SUV (SUV_{peak}) assessment. When compared to maximum SUV (SUV_{max}), SUV_{peak} has shown to be more robust, due to the fact that it is less adversely affected by noise. Another issue might be seen in the use of a quantification method which is based on bodyweight normalized SUV. As presented in section 2.1.5, this SUV index has a relatively simple experimental protocol, and for that reason tends to be considerable inaccurate when it comes to comparison of quantification results from different patients. However, integration of algorithms for body surface area or lean body mass normalized SUV has its pitfalls too. The use of bodyweight normalized SUV is so widespread in clinical practice that the required additional parameters for estimation of body surface area or lean body mass SUVs are often missing in the PET DICOM data. Hence, there always remains the risk that such image data becomes incompatible with the software or, even worse, quantification might deliver false results. A possible workaround for this problem could be obtained by offering a fallback solution in form of bodyweight normalized SUV computation in case that required parameters for execution of the more sophisticated methods are missing.

Considering the strategy for further development of the software, it would be best to continue with enhancement of the data structure to support SUV_{peak} assessment and to develop and integrate a new segmentation tool, which is capable to leverage this enhanced implementation. Next, integration of body surface area or lean body mass SUV computation methods should be considered in order to be able to provide more reliable quantification results. With these enhancements, the software can be brought to a level of functionality akin to the one offered by commercial PET analysis software.

For further support of external developers, an enhancement of the available interfaces for integration of new algorithms should be taken into consideration. While the plugin architecture for extension of segmentation tools is provided by the implementation of Slicer's Editor module, the interface for extension of quantification algorithms was implemented as a part of this work. External developers can submit

specific enhancement requests through the Slicer bug tracking system or by using the open mailing lists provided by the Slicer developer community. In summary, it can be said that the majority of the missing features and usability issues that were revealed during the evaluation of this work, can be implemented or fixed with relatively little effort due to the well-elaborated architecture of the developed software.

5.1 Conclusion

In conclusion, an extensible, free open source software tool for longitudinal analysis of tumor changes in PET imaging has been developed in this project. With its currently available default configuration, this tool covers the basic requirements for PET quantification. Development and integration of more advanced PET segmentation and quantification methods can be implemented in the future by leveraging the extensible architecture of the software.

Bibliography

[1] R. Boellaard, “Standards for PET image acquisition and quantitative data analysis,” *Journal of nuclear medicine : official publication, Society of Nuclear Medicine*, vol. 50 Suppl 1, p. 11S–20S, May 2009.

[2] H. S. Huang, “Anatomy of SUV,” *Nuclear Medicine and Biology*, vol. 27, no. 7, pp. 643–646, 2000.

[3] K. Zasadny and L. Wahl, “Standardized uptake values of normal tissues at PET with 2-[fluorine-18]-fluoro-2-deoxy-D-glucose: variations with body weight and a method for correction.,” *Radiology*, vol. 189, no. 3, pp. 847–850, 1993.

[4] Y. Sugawara, K. R. Zasadny, a W. Neuhoff, and R. L. Wahl, “Reevaluation of the standardized uptake value for FDG: variations with body weight and methods for correction,” *Radiology*, vol. 213, no. 2, pp. 521–5, Nov. 1999.

[5] R. J. Hicks, “The SUV and FLT PET: a tasty alphabet soup or a dog’s breakfast?,” *Leukemia & lymphoma*, vol. 48, no. 4, pp. 649–52, Apr. 2007.

[6] C. K. Kim, N. C. Gupta, B. Chandramouli, A. Alavi, and P. Population, “Standardized Uptake Values of FDG: Body Surface Area Correction is Preferable to Body Weight Correction,” *The Journal of Nuclear Medicine*, vol. 35, no. 1, pp. 164–167, 1994.

[7] M. Vanderhoek, S. B. Perlman, and R. Jeraj, “Impact of the definition of peak standardized uptake value on quantification of treatment response,” *Journal of nuclear medicine : official publication, Society of Nuclear Medicine*, vol. 53, no. 1, pp. 4–11, Jan. 2012.

[8] G. Tomasi, F. Turkheimer, and E. Aboagye, “Importance of Quantification for the Analysis of PET Data in Oncology: Review of Current Methods and Trends for the Future,” *Molecular imaging and biology MIB the official publication of the Academy of Molecular Imaging*, vol. 13, no. 2, pp. 1–16, 2011.

[9] R. N. Gunn, S. R. Gunn, and V. J. Cunningham, “Positron emission tomography compartmental models,” *Journal of cerebral blood flow and metabolism : official journal of the International Society of Cerebral Blood Flow and Metabolism*, vol. 21, no. 6, pp. 635–52, Jun. 2001.

[10] J. Logan, “Graphical analysis of PET data applied to reversible and irreversible tracers,” *Nuclear medicine and biology*, vol. 27, no. 7, pp. 661–70, Oct. 2000.

[11] V. J. Cunningham and T. Jones, “Spectral analysis of dynamic PET studies,” *Journal of cerebral blood flow and metabolism : official journal of the International Society of Cerebral Blood Flow and Metabolism*, vol. 13, no. 1, pp. 15–23, Jan. 1993.

[12] H. Jadvar, “PET Physics and Instrumentation,” in in *Clinical PET and PET/CT*, London: Springer London, 2005, pp. 1–43.

[13] T. B. Lynch, “Basic Physics,” in in *PET/CT in Clinical Practice*, London: Springer London, 2007, pp. 204–214.

[14] M.-J. Martinez, S. I. Ziegler, and T. Beyer, “PET and PET/CT: Basic Principles and Instrumentation,” in in *PET in Oncology*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–23.

[15] J. Langner, “PET Schema.” [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/c/c1/PET-schema.png. [Accessed: 01-Mar-2013].

[16] D. L. Bailey, “Data Acquisition and Performance Characterization in PET,” in in *Positron Emission Tomography Basic Sciences*, London: Springer London, 2005, pp. 41–62.

[17] M. Defrise, P. E. Kinahan, and C. J. Michel, *Image Reconstruction Algorithms in PET*. London: Springer London, 2005.

[18] T. Beyer, D. W. Townsend, T. Brun, P. E. Kinahan, M. Charron, R. Roddy, J. Jerin, J. Young, L. Byars, and R. Nutt, “A combined PET/CT scanner for clinical oncology,” *Journal of nuclear medicine : official publication, Society of Nuclear Medicine*, vol. 41, no. 8, pp. 1369–79, Aug. 2000.

[19] T. E. Nichols, J. Qi, and R. M. Leahy, “Continuous Time Dynamic PET Imaging Using List Mode Data,” no. Ml, pp. 98–111, 1999.

[20] H. W. Atabe, Y. I. Koma, Y. K. Imura, M. N. Aganawa, and M. S. Hidahara, “PET kinetic analysis — compartmental model,” *Annals of Nuclear Medicine*, vol. 20, no. 9, pp. 583–588, 2006.

[21] N. E. M. Association, “Digital Imaging and Communications in Medicine (DICOM). Part 1. Introduction and overview.” p. 5, 2011.

[22] O. S. Pianykh, *Digital Imaging and Communications in Medicine (DICOM)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–68.

[23] N. E. M. Association, “Digital imaging and communications in medicine (DICOM) Part 3: Information object definitions.” p. 46, 2011.

[24] N. E. M. Association, “Digital imaging and communications in medicine (DICOM) Part 6: Data dictionary,” *NEMA PS*. 2011.

[25] N. E. M. Association, “Digital Imaging and Communications in Medicine (DICOM) Part 5: Data Structures and Encoding.” pp. 23–31, 2011.

[26] R. Salgado, T. Mulkens, P. Bellinck, and J. L. Termote, “Volume rendering in clinical practice. a pictorial review,” *JBR-BTR: Journal Belge de Radiologie - Belgisch Tijdschrift voor Radiologi)*, vol. 86, no. 4, pp. 215–20, 2003.

[27] P. Van Ooijen, “Noninvasive coronary imaging using electron beam CT: surface rendering versus volume rendering,” *American Journal of Roentgenology*, no. January, pp. 223–226, 2003.

[28] DKFZ, “Leber OP Planung - Forschungsprojekte der Abteilung Medizinische und Biologische Informatik.” [Online]. Available: http://www.dkfz.de/de/mbi/projects/leber.html. [Accessed: 19-Jun-2013].

[29] W. J. Schroeder, K. Martin, and B. Lorensen, *The visualization toolkit*. [Clifton Park, NY]: Kitware, 2006, p. XVI, 512 S.

[30] J. M. Fitzpatrick and M. Sonka, “Image Registration,” in in *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis*, Society of Photo Optical, 2000, pp. 447–513.

[31] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[32] B. Van Ginneken and A. Frangi, “Active shape model segmentation with optimal features,” *IEEE Transactions on Medical Imaging*, vol. 21, no. 8, pp. 924–933, 2002.

[33] L. Ibáñez and W. J. Schroeder, Eds., *The ITK software guide*. [New York]: Kitware, 2005, p. XXXII, 787 S.

[34] NA-MIC, “National Alliance for Medical Image Computing.” [Online]. Available: http://www.na-mic.org/. [Accessed: 19-Apr-2013].

[35] S. Pieper, B. Lorensen, W. Schroeder, and R. Kikinis, “The NA-MIC Kit : ITK , VTK , Pipelines , Grids and 3D Slicer as An Open Platform for the Medical Image Computing Community 5) National Alliance for Medical Image Computing,” pp. 698–701, 2006.

[36] “NA-MIC-Kit,” 2013. [Online]. Available: http://www.na-mic.org/Wiki/index.php/NA-MIC-Kit. [Accessed: 25-Feb-2013].

[37] The Common Toolkit, “Readme.” [Online]. Available: https://github.com/commontk/CTK#readme. [Accessed: 22-Feb-2013].

[38] Qt Project, “All Classes.” [Online]. Available: http://qt-project.org/doc/qt-4.8/classes.html. [Accessed: 12-Apr-2013].

[39] Nokia, “Using the Meta-Object Compiler (moc).” [Online]. Available: http://harmattan-dev.nokia.com/docs/library/html/qt4/moc.html. [Accessed: 12-Apr-2013].

[40] Qt Project, “Signals & Slots.” [Online]. Available: http://qt-project.org/doc/qt-4.8/signalsandslots.html. [Accessed: 12-Apr-2013].

[41] P. Mildenberger, M. Eichelberg, and E. Martin, “Introduction to the DICOM standard,” *European radiology*, vol. 12, no. 4, pp. 920–7, Apr. 2002.

[42] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, J. Buatti, S. Aylward, J. V Miller, S. Pieper, and R. Kikinis, “3D Slicer as an image computing platform for the Quantitative Imaging Network,” *Magnetic resonance imaging*, vol. 30, no. 9, pp. 1323–41, Nov. 2012.

[43] V. Vezhnevets, “‘GrowCut’ - Interactive Multi-Label N-D Image Segmentation By Cellular Automata,” 2004.

[44] J. Nielsen, *Usability engineering*. San Francisco, Calif. [u.a.]: Kaufmann, 1993, pp. 23–37, 155–163.

[45] J. Nielsen, “Usability 101: Introduction to Usability.” [Online]. Available: http://www.nngroup.com/articles/usability-101-introduction-to-usability/. [Accessed: 20-Jun-2013].

[46] A. E. Carpenter, L. Kamentsky, and K. W. Eliceiri, “A call for bioimaging software usability,” *Nature methods*, vol. 9, no. 7, pp. 666–70, Jul. 2012.

[47] R. Mack, J. N. N, H. Laboratories, P. Brooks, B. North, and R. J. H. Laboratories, “Usability Inspection Methods,” 1993.

[48] J. Zhang, T. R. Johnson, V. L. Patel, D. L. Paige, and T. Kubose, “Using usability heuristics to evaluate patient safety of medical devices,” *Journal of Biomedical Informatics*, vol. 36, no. 1–2, pp. 23–30, Feb. 2003.

[49] S. Lauesen, *User interface design*. Harlow ; Munich [u.a.]: Pearson/Addison-Wesley, 2005, p. XVIII, 604 S.

[50] W. W. Royce, “Managing the Development of Large Software Systems: Concepts and Techniques,” in *IEEE WESCON*, 1970, no. August, pp. 1–9.

[51] B. W. Boehm, “A spiral model of software development and enhancement,” *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[52] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods,” *VTT Publications*, pp. 11–12, 2002.

[53] K. Beck, *Extreme programming explained*. Reading, Massachusetts: Addison-Wesley, 2000, p. XXI, 190 S.

[54] K. Schwaber, *Agile project management with Scrum*. Redmond, Washington: Microsoft Press, 2004, p. XIX, 163 S.

[55] J. Ludewig and H. Lichter, *Software-Engineering*. Heidelberg: dpunkt-Verlag, 2007, pp. 209–10, 377–78.

[56] M. Chemuturi, “Elicitation and Gathering of Requirements,” in in *Requirements Engineering and Management for Software Development Projects*, New York, NY: Springer Berlin Heidelberg, 2013, pp. 33–54.

[57] S. Pieper, M. Halle, and R. Kikinis, “3D SLICER,” in *Proceedings of the 1st IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2004, pp. 632–635.

[58] NA-MIC, “2012 Summer Project Week.” [Online]. Available: http://www.na-mic.org/Wiki/index.php/2012_Summer_Project_Week. [Accessed: 04-Apr-2013].

[59] NA-MIC, “Project Events.” [Online]. Available: http://wiki.na-mic.org/Wiki/index.php/Project_Events. [Accessed: 04-Apr-2014].

[60] Slicer Wiki, “Modules.” [Online]. Available: http://www.slicer.org/slicerWiki/index.php/Documentation/4.2/Developers/Modules. [Accessed: 10-Apr-2013].

[61] R. L. Wahl, H. Jacene, Y. Kasamon, and M. A. Lodge, “From RECIST to PERCIST: Evolving Considerations for PET response criteria in solid tumors.,” *Journal of nuclear medicine : official publication, Society of Nuclear Medicine*, vol. 50 Suppl 1, no. Suppl 1, p. 122S–50S, May 2009.

[62] P. Mercea, “Workflow Demonstration: Longitudinal PET/CT Analysis Extension for 3D Slicer 4.” [Online]. Available: https://www.youtube.com/watch?v=O7Di6UZDsf0. [Accessed: 29-Jun-2013].

A Installation Guide

This section presents the installation process of the Longitudinal PET/CT Analysis module using the Extension Manager in 3D Slicer 4. As a prerequisite for this installation, 3D Slicer 4 (version 4.2 or higher) needs to be installed first on a Windows, Linux or Mac OSX platform. Stable releases and nightly build installer packages are available via the Slicer download page[34].

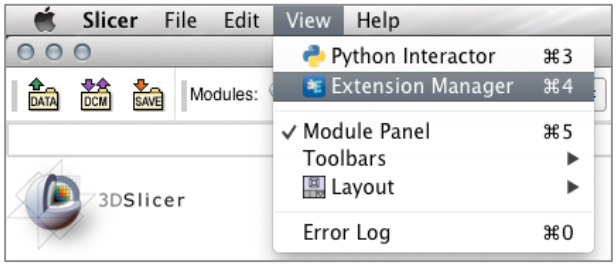


Fig. A.1: Opening Slicer's Extension Manager from the menu bar.

In a first step, Slicer's Extension Manager has to be opened using the menu bar of the main application. It is located in the "View" menu (see Fig. A.1). The user interface of the Extension Manager dialog is divided in two different tabs. For installation of new extensions, the "Install Extensions" tab needs to be opened. In there, all extensions available at the timepoint of opening the Extension Manager are presented. Installation of the Longitudinal PET/CT Analysis extension is initiated by clicking on the "Install" button located underneath its icon (see Fig. A.2). Thereupon, the installation is performed automatically in the background. When successfully installed, the application has to be restarted in order to be able to access the functionality of the newly installed extension.

[34] 3D Slicer 4 download: http://download.slicer.org/

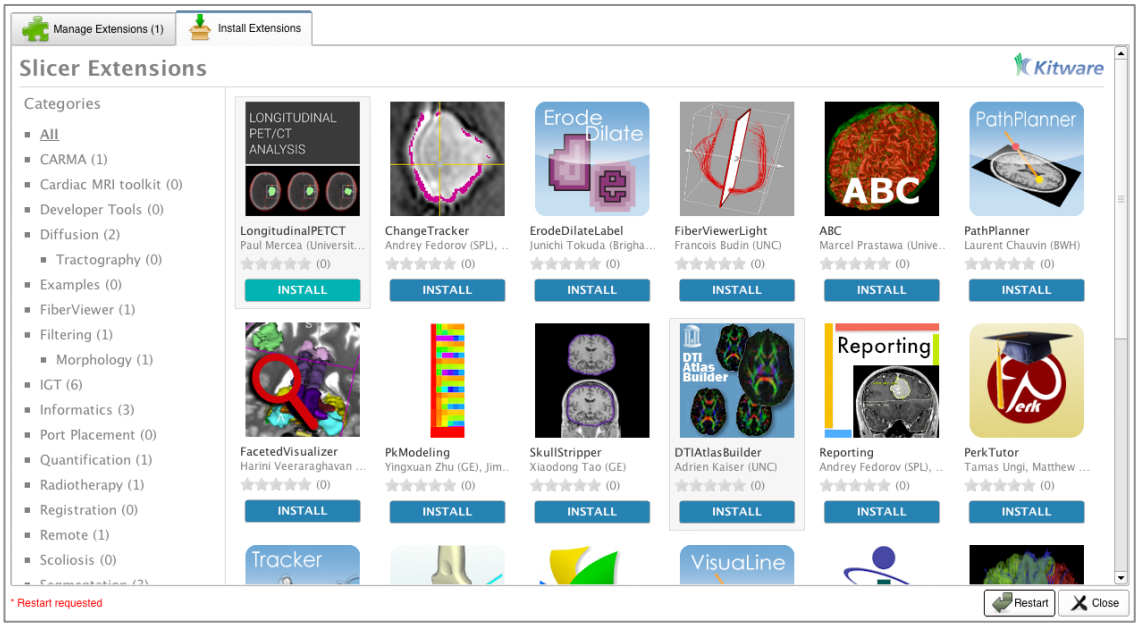


Fig. A.2: 3D Slicer 4 Extension Manager “Install Extensions” view.

In addition to the possibility for installation of the Longitudinal PET/CT Analysis extension using Slicer’s Extension Manager, the source code of this project is also available online and can be used to build the binary files of the software manually. The source files can be found at the web-based hosting service *github*[35]. The exact location of the repository used in this work is: https://github.com/paulcm/LongitudinalPETCT.

[35] GitHub: https://github.com/

B Documentation

Documentation for the Longitudinal PET/CT Analysis extension for 3D Slicer 4 is available in the Slicer Wiki[36]. On this web page, a short description of the software, use cases and an overview of the different GUI panels and their use are given.

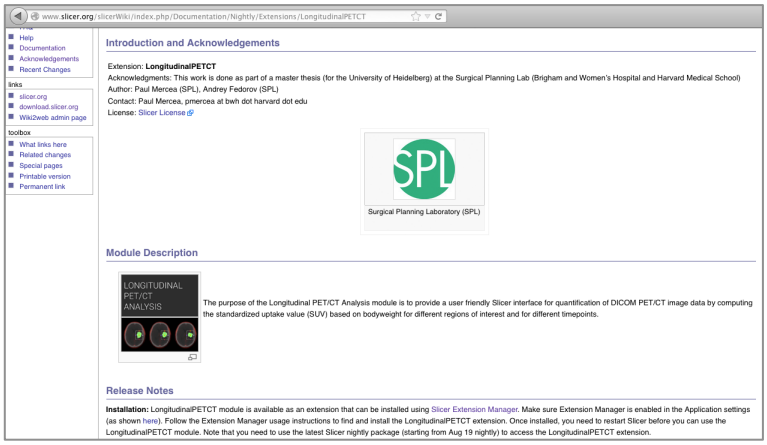


Fig. B.1: Slicer Wiki page for the Longitudinal PET/CT Analysis extension

Moreover, a screencast demonstrating the workflow of the Longitudinal PET/CT Analysis extension is also available online[37].

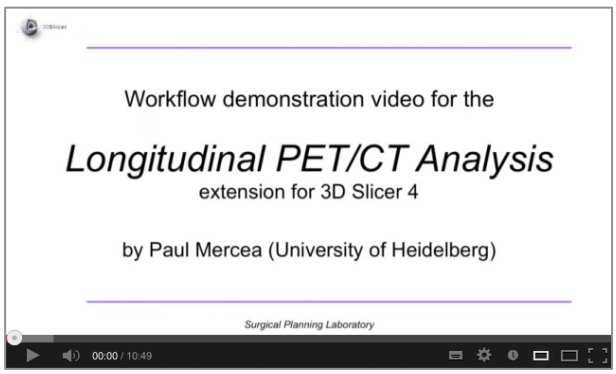


Fig. B.2: Screenshot of the workflow demonstration video.

[36] Longitudinal PET/CT Extension Documentation: http://slicer.org/slicerWiki/index.php/Documentation/Nightly/Extensions/LongitudinalPETCT

[37] Longitudinal PET/CT Extension workflow demonstration video: http://youtube.com/watch?v=O7Di6UZDsf0&feature=youtu.be

C UML Class Diagrams

Fig. C.1: Detailed UML class diagram of the Model classes containing all attributes and public operations.

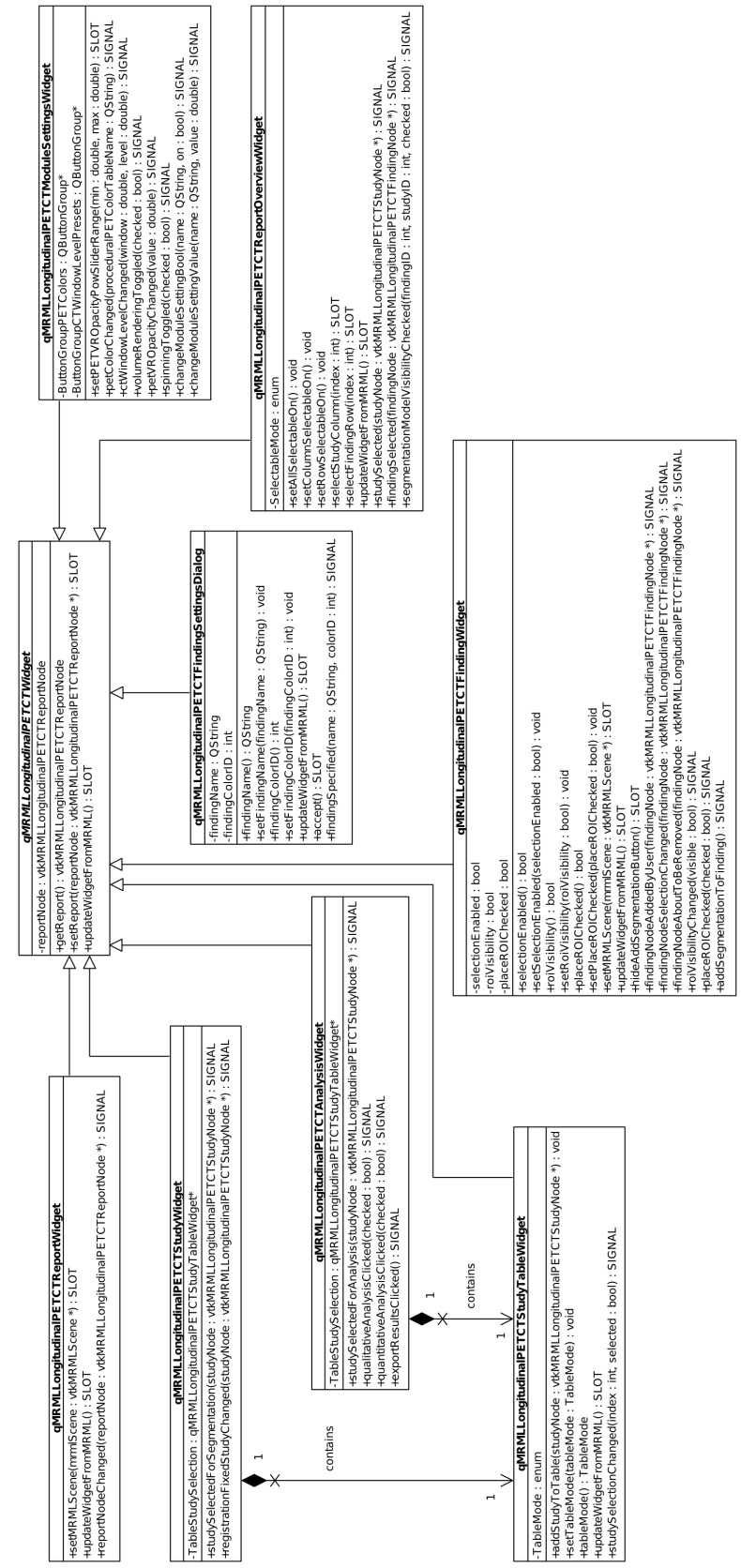


Fig. C.2: Detailed UML class diagram of the View classes containing signal, slot and other public operations.

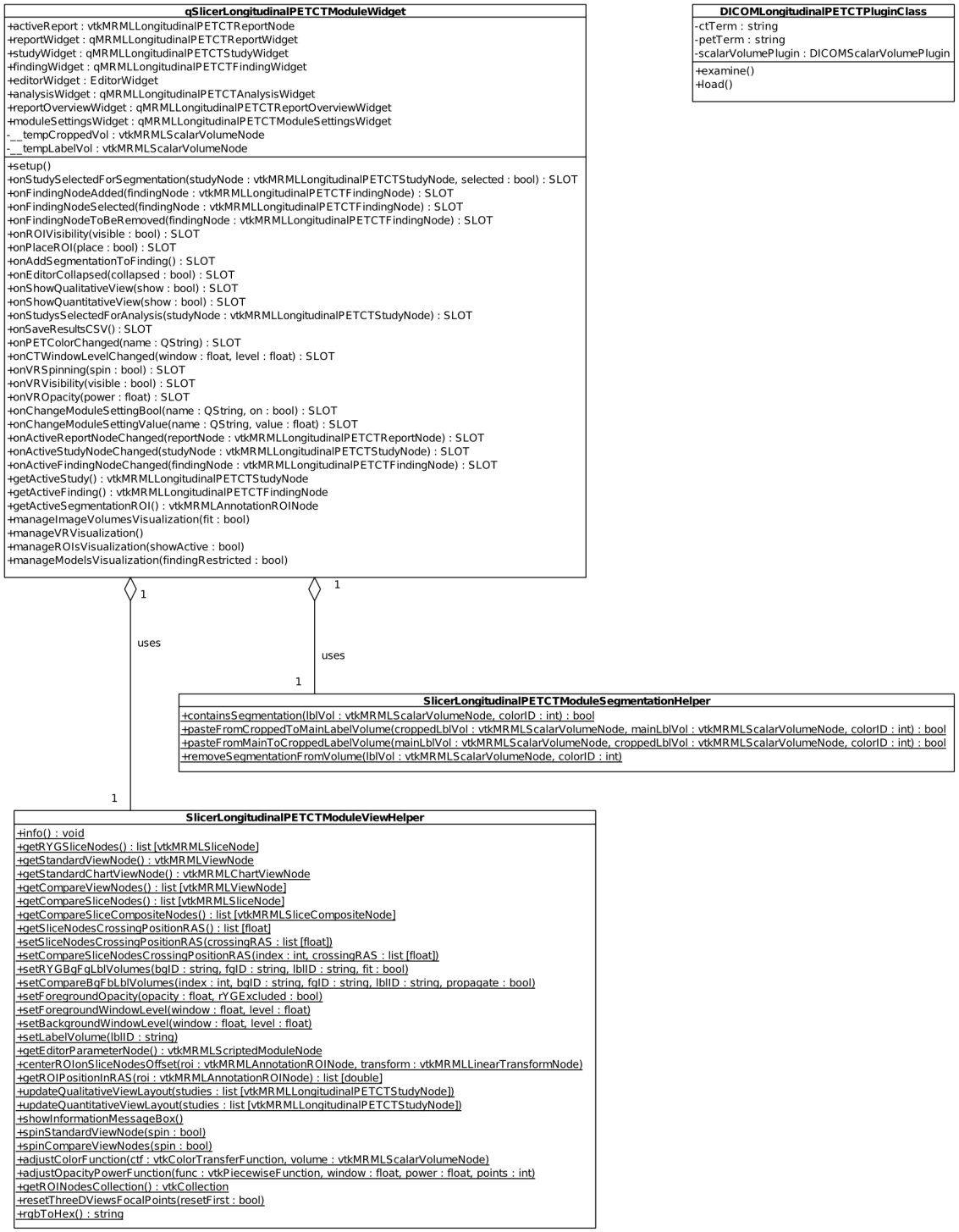
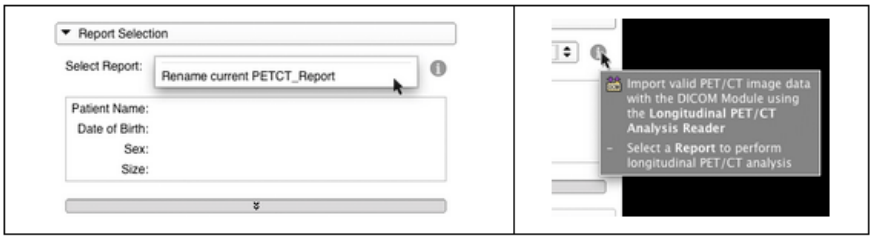


Fig. C.3: Detailed UML class diagram of the Controller classes, composed of the main Logic class, its helper classes and the standalone DICOM plugin class.

D Accompanying Questionnaire for Heuristic Evaluation

1. Importing a Longitudinal PET/CT Report from DICOM DB



Q1: No PETCT_Report is available for selection or creation until it has been imported with the DICOM Module. Does the Report Selection panel's information tooltip give enough information about what has to be done to be able to select a Report ?

( ) Yes, the description is very helpful explaining how to get a Report imported

( ) No, it is not clear how to proceed with the workflow with the information given by the tooltip

others:_____

Q2: Which of the following points is the most accurate in describing what you think about the information tooltip concept ?

( ) The information tooltips in each panel of the module can help unexperienced users to understand the functionality/features of a panel with just a simple mouse over.

( ) The information given by the information tooltips is useful but a button opening a pop-up dialog with this information is more intuitive.

( ) The information tooltips are disturbing and should not be placed in the panels. Documentation does not belong directly in the user interface.

others:_____

1

Fig. D.1: First page of the questionnaire accompanying the heuristic evaluation.

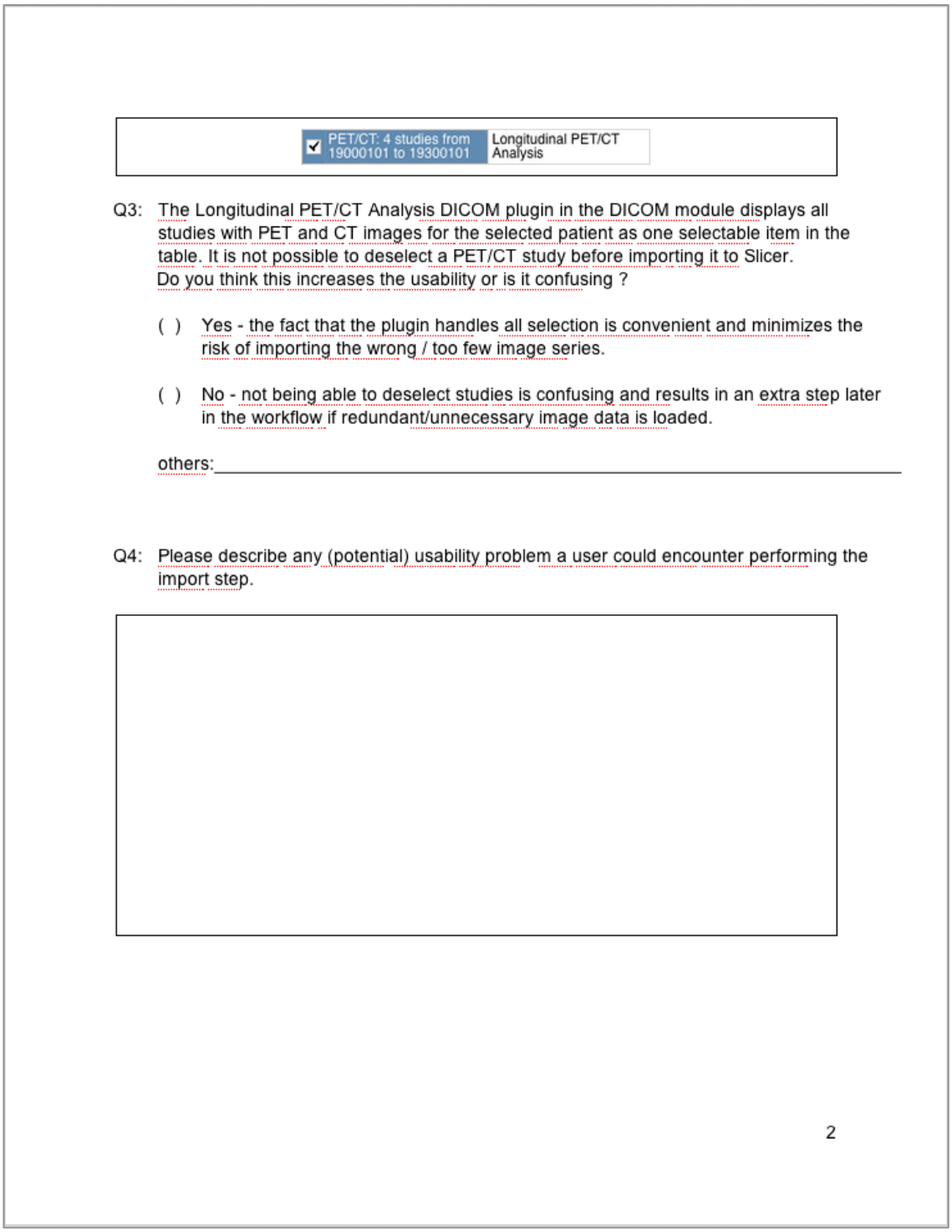
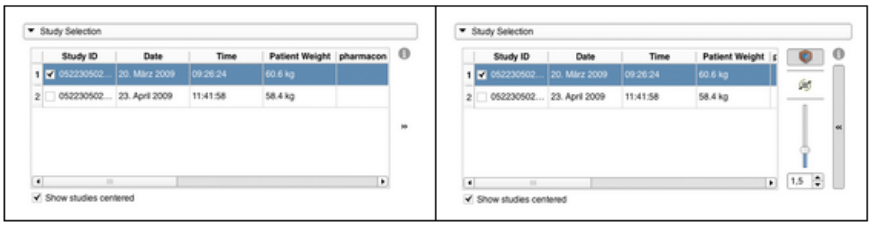


Fig. D.2: Second page of the questionnaire accompanying the heuristic evaluation.

2. Selecting studies for the workflow



Q1: The expandable advanced settings panel for the volume rendering settings is hidden by default to keep the user interface simple. Do you think this is a good approach ?

( ) Yes - a simple interface design increases usability (especially for unexperienced users) and experienced users can easily expand the panel and adjust the settings.

( ) No - unexperienced users might overlook the expand button and not be able to disable volume rendering / or spinning which on slower computers might be an important setting

others: _____

Q2: Once the first study has been selected for the workflow, volume rendering of its PET volume is initialized and displayed in the 3D view. Since volume rendering can be a resource hog (especially on CPU), should it be disabled by default to not confuse users with the sudden increase of response time of the system (although clinicians in the field of nuclear medicine are used to have the PET image data visualized as a spinning 3D volume) ?

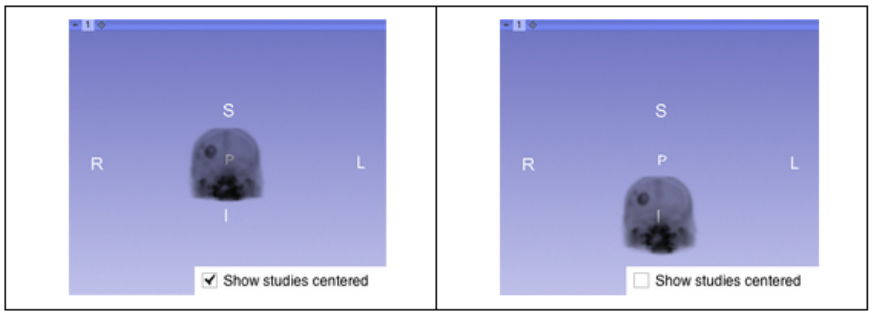
( ) Yes - a sudden change in the responsiveness of the system is very disturbing and should be avoided. In the worst case a user might think the software crashed. If the user has the right hardware he can activate volume rendering manually and should be able to save this setting for future use of the module.

( ) No - functionality should not be restricted by default. It is sufficient if the user has the possibility to disable functionalities if his hardware does not support them.

others: _____

3

Fig. D.3: Third page of the questionnaire accompanying the heuristic evaluation.



Q3: The Longitudinal PET/CT extension does not offer registration but in the Study Selection panel the user can select whether the PET and CT volumes of a Study should be centered at the origin of the RAS instead being placed at the RAS position defined by the DICOM files. This is a convenience setting to help users to not loose the orientation in the 3D viewer. Do you think this is a useful feature ?

( ) Yes

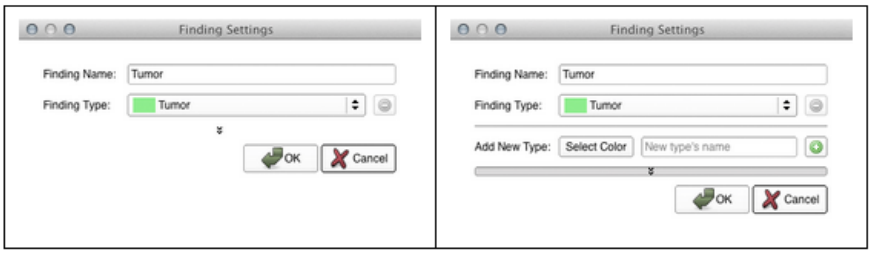
( ) No - reason: _____

Q4: Please describe any (potential) usability problem a user could encounter selecting/deselecting studies for the workflow.

4

Fig. D.4: Forth page of the questionnaire accompanying the heuristic evaluation.

3. Creating/editing a Finding



Q1: After selecting "Create new Finding" the Finding Settings dialog pops up offering the user an input panel to set the name and the type of the Finding (and also an advanced settings panel for creating new types). Should this input fields rather be in the panel itself than in an extra popup dialog ?

( ) Yes - reason: _____

( ) No

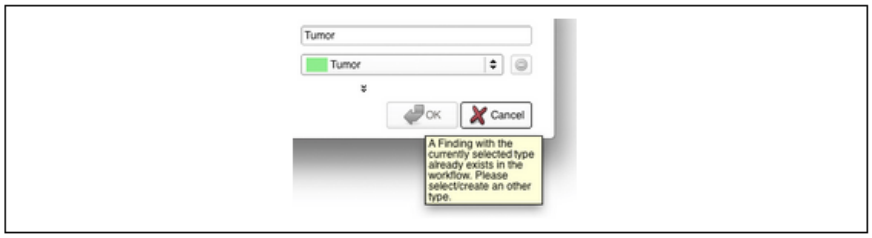
Q2: Selecting a different Finding type in the Finding Settings dialog after creation of a new Finding will automatically rename the Finding regarding the selected type. Do you think this is a useful or rather confusing feature ?

( ) useful

( ) confusing - reason:_____

5

Fig. D.5: Fifth page of the questionnaire accompanying the heuristic evaluation.



Q3: The Longitudinal PET/CT Analysis extension does not allow the usage of two Findings of the same type. To avoid creation of a Finding with an already existent type the OK button in the Finding Settings dialog is disabled and a tooltip explaining the reason is shown when moving the mouse over the button. The button gets enabled as soon as another type is selected by the user. What do you think is more user friendly ?

( ) Disabled OK button with tooltip explaining the reason of disability

( ) Pop-up warning message dialog when clicking OK explaining that another type has to be selected and returning back to Dialog.

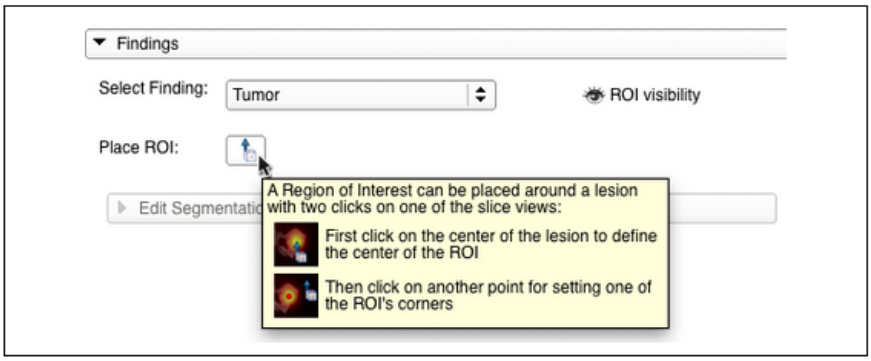
( ) other suggestion: _____

Q4: Please describe any (potential) usability problem a user could encounter creating/editing a Finding with the Finding Settings Dialog.

6

Fig. D.6: Sixth page of the questionnaire accompanying the heuristic evaluation.

4. Placing ROI's



Q1: The Longitudinal PET/CT Analysis extension utilizes Slicer's ROI placement functionality for the definition of regions of interest around lesions. Since unexperienced users might not exactly know how to use this function an illutstrated tooltip explains the steps to perform the placement. How do you rate the usability of this tooltip explanation approach:

( ) not necessary - Slicer's ROI placing is easy to learn
( ) could be usefulbut tooltips should be short and not include extented documentation
( ) useful - but the images are too small / explanation is not precise enough
( ) very useful - all special features should be explained by illustrated tooltips

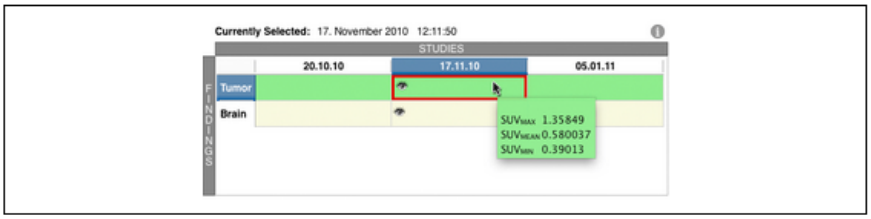
Q2: If you think that the 2 click ROI placement is not intuitive enough do you have any suggestions/ideas for an improved ROI placement procedure (e.g. one click for placing an ROI with fixed size) ?

_____
_____
_____

7

Fig. D.7: Seventh page of the questionnaire accompanying the heuristic evaluation.

5. Keeping track with the Report Overview Table



Q1: Please rate the usability of the Report Overview Table:

not useful (confusing)		useful (but too sophisticated)		very useful (easy to use)
[ ]	[ ]	[ ]	[ ]	[ ]

Q2: What do you think about the feature of selecting the Finding/Study currently visualized by simply clicking on a table cell in the Report Overview Table ?

( ) useful - and easy to use

( ) confusing - selection of Findings and Studies should be done by selecting from dropdown box (combobox)

other thoughts :_____

Q3: Visibility of a segmentation's model representation can be enabled/disabled directly from the visibility button in the table cell. Would you say the placement for these visibility buttons in the table makes sense or is there a better place for them in the module ?

( ) Yes - it makes sense

( ) No - they should rather be placed in _____
_____
_____

Q4: Please describe any (potential) usability problem a user could encounter using the Report Overview Table.

8

Fig. D.8: Eighth page of the questionnaire accompanying the heuristic evaluation.

Note: The screenshots used in this questionnaire may contain some minor differences when compared to the GUI widgets presented in this work. This is due to the fact that the screenshots were taken from a preliminary development version of the extension. However, these differences do not affect the validity of the heuristic evaluation (see section 3.4.3) since only feedback from the free text answers were used in that one.